

Introduction to Deep Learning for Geoscience

Ronan Fablet (ronan.fablet@ilt-atlantique.fr)

Course on “Data Science for Geosciences”, DSG’2020, Toulouse, January 2020

This course focuses on "**an introduction to deep learning**", not just "**how to use**" deep learning in practice.

Resources

- **Book: Deep Learning**

Goodfellow, Bengio, Courville, MIT Press

Online version <http://www.deeplearningbook.org/>

- **Online course by Andrew Ng (Stanford/Baidu)**

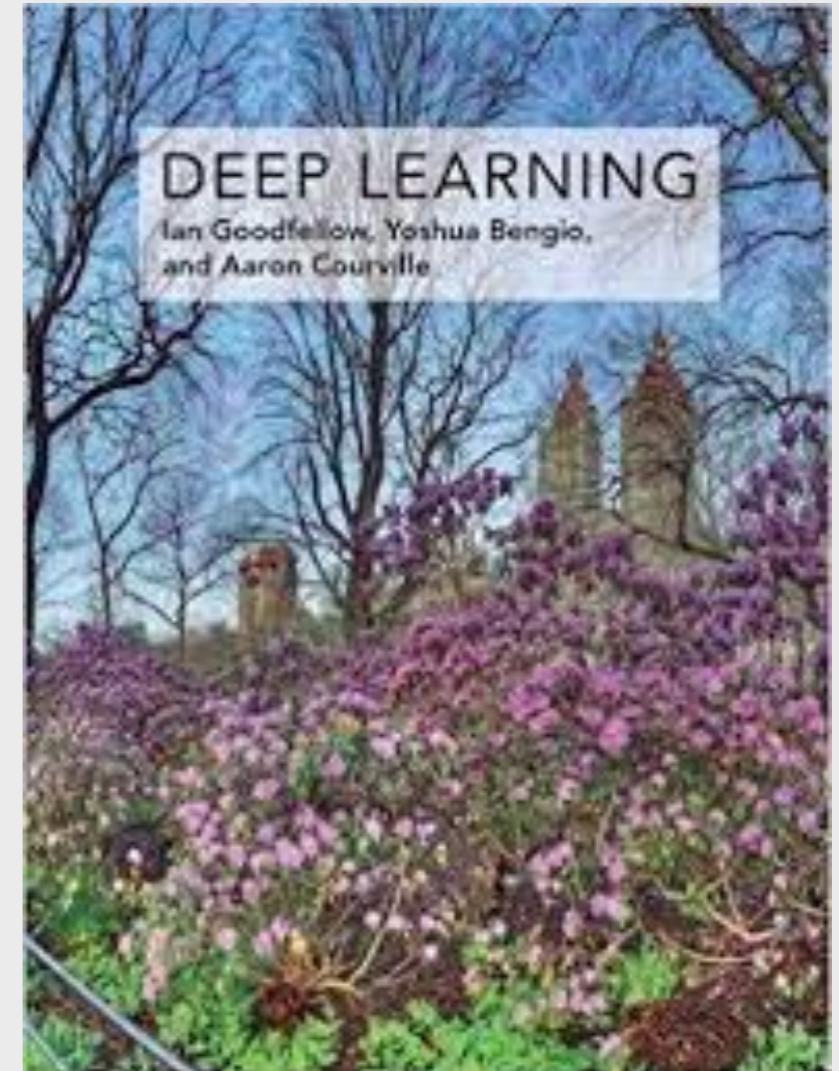
Youtube: [link](#)

Online course on Coursera: [link](#)

- **Review paper: Deep learning in neural networks by**

J. Schmidhuber

pdf: [link](#)



Roadmap

- What is deep learning ?
- What does (machine) learning mean ?
- What are Neural Networks ?
- What are the key components of DL models ?
- How to implement DL models using Keras ?
- A (very) short tour of the NN zoo

What is Deep Learning?

What is Deep Learning?

Artificial Intelligence

Machine Learning

Random
Forest

SVM

Nearest-
Neighbour

Deep
Learning

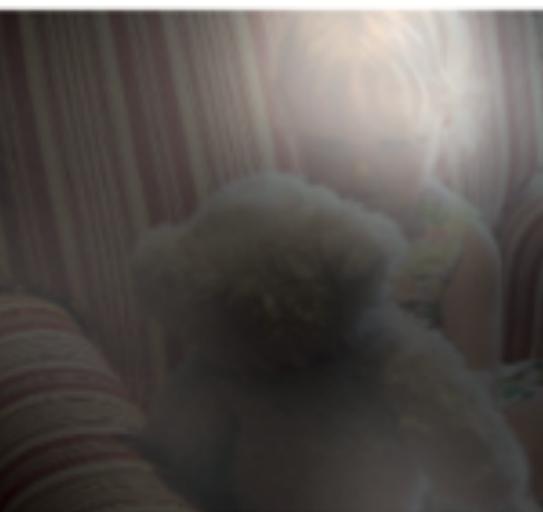
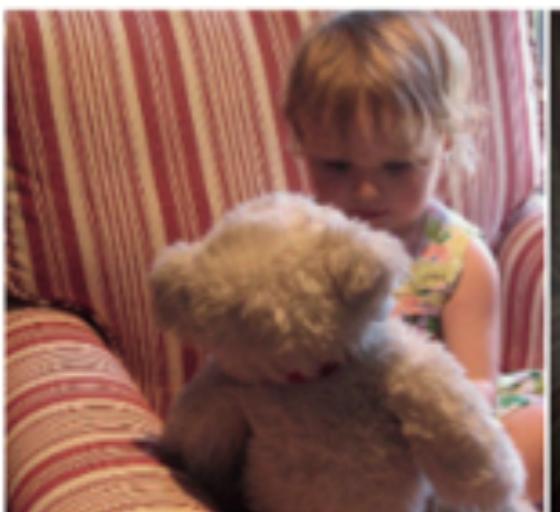
**Deep learning has rapidly become the state-of-art-framework
for a wide range of applications in computer vision, natural
language processing, signal processing...**

From Image to Text (image captioning)



A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.



A little **girl** sitting on a bed with a **teddy bear**.



A group of **people** sitting on a boat in the water.

Automatic Translation

The image shows two side-by-side screenshots of automatic translation tools. The top half displays Google Translate's interface, where a French sentence "Ce cours sur l'apprentissage profond est génial." is translated into English as "This deep learning course is great." A red stamp with the text "Early 2017" is overlaid on the right side of the Google Translate screenshot. The bottom half displays DeepL's interface, showing the same French sentence being translated into English as "This course on deep learning is great." Both interfaces include language selection dropdowns, a word count indicator (48/5000), and a "Suggérer une modification" (Suggest edit) button.

Google

Traduction

Désactiver la traduction instantanée

Anglais Français Arabe Déterminer la langue

Traduire

Ce cours sur l'apprentissage profond est génial.

This deep learning course is great.

48/5000

Suggérer une modification

DeepL

Translate from FRENCH (detected)

Translate into ENGLISH

Ce cours sur l'apprentissage profond est génial.

This course on deep learning is great.

To look up words in the dictionary, just click on them.

Early 2017

and many more.....

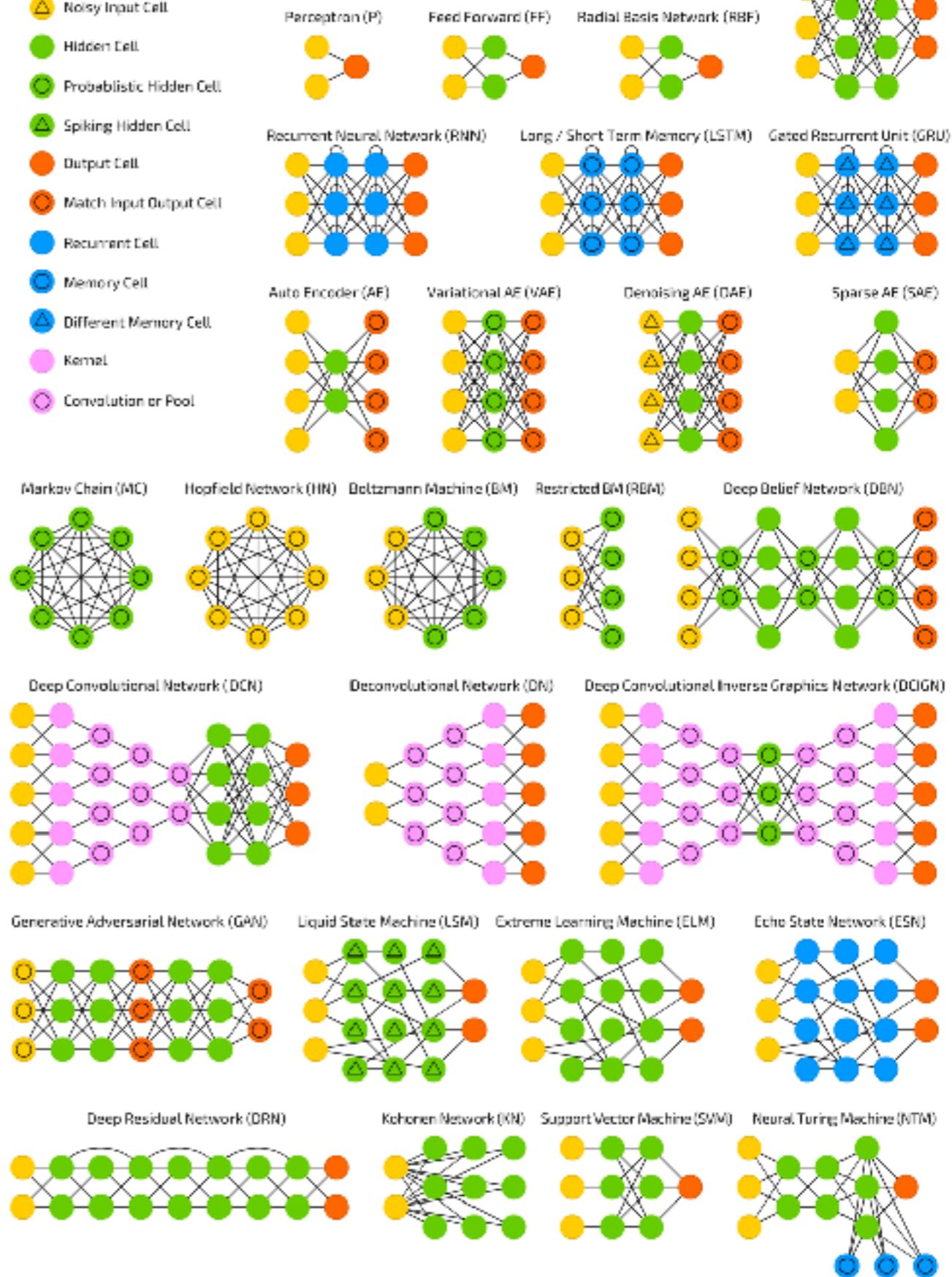
Deep learning relies on neural networks, which are not new...

Artificial neural networks date back to the 60's and were popular in the 80's.

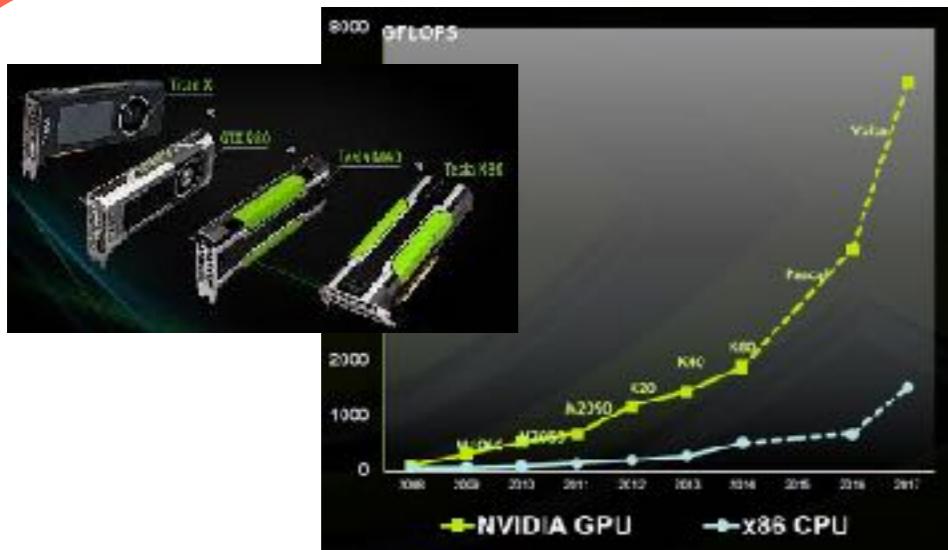
A mostly complete chart of Neural Networks

©2016 Fjodor van IJken - asimovinstitute.org

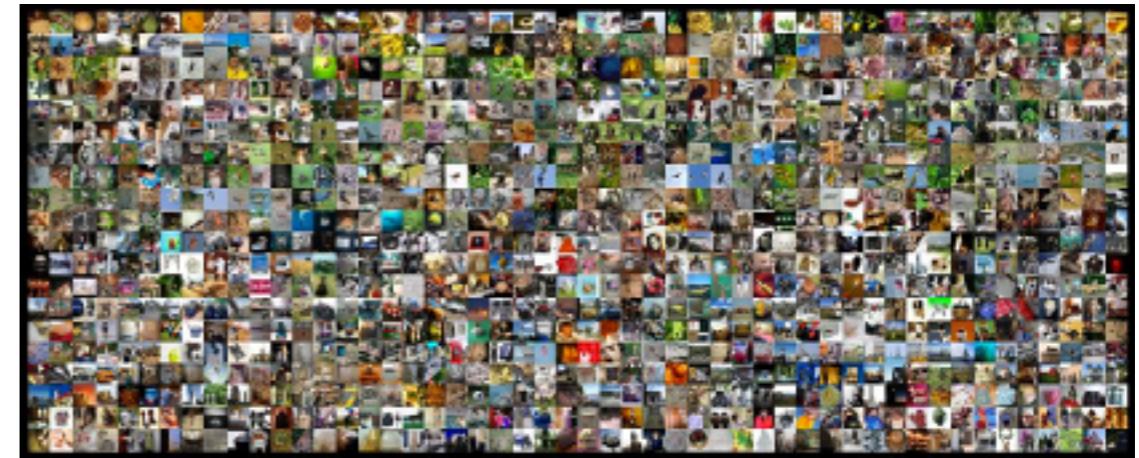
- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool



Key reasons for DL emergence



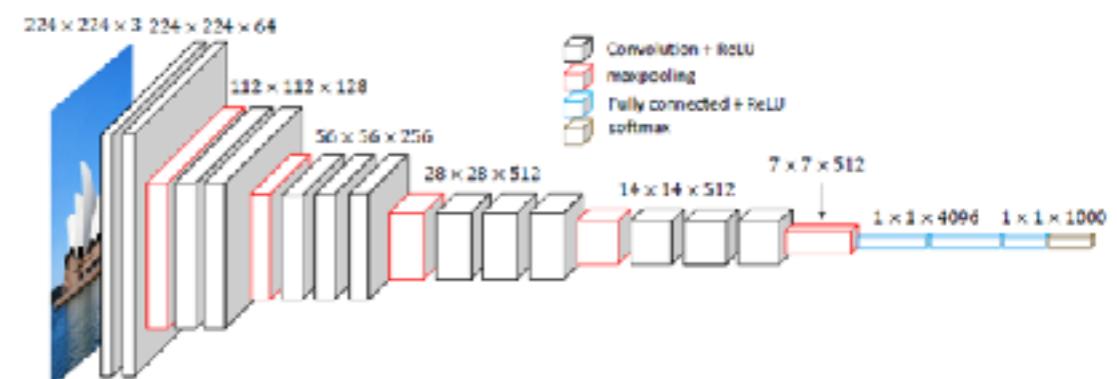
High-performance computing (GPU)



Large annotated dataset (> 1M)



Efficient & easy-to-use libraries



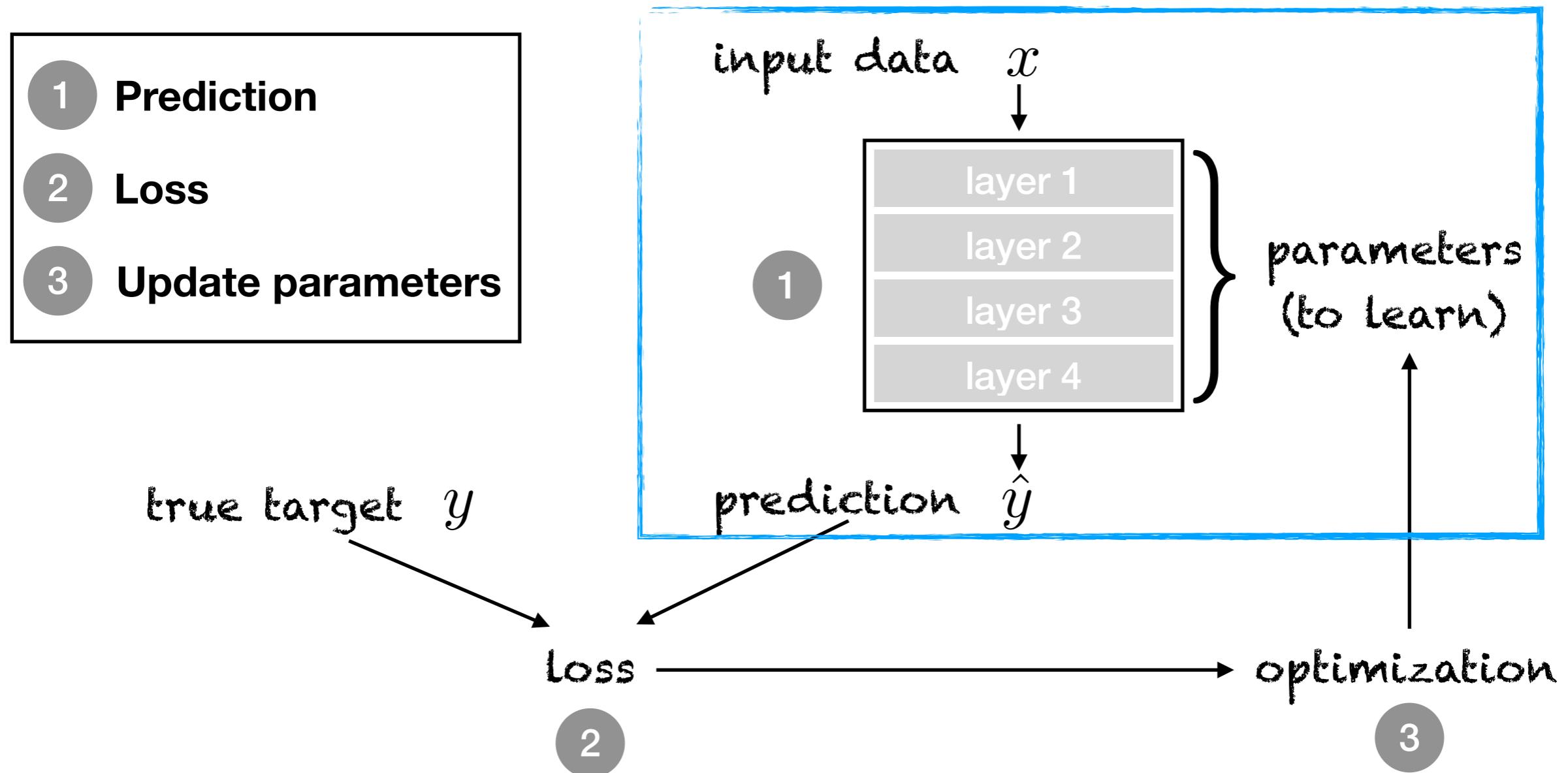
End-to-end learning

**WHAT DOES (MACHINE)
LEARNING MEAN ?**

Machine/Deep Learning

Definition (wikipedia): Machine learning algorithms build a **mathematical model** based on sample data, known as "**training data**", in order to make predictions or decisions without being explicitly programmed to perform the task.

Machine learning



Machine/Deep Learning

Mathematical formulation: Learning comes to minimising some loss function given w.r.t. model parameters and training data

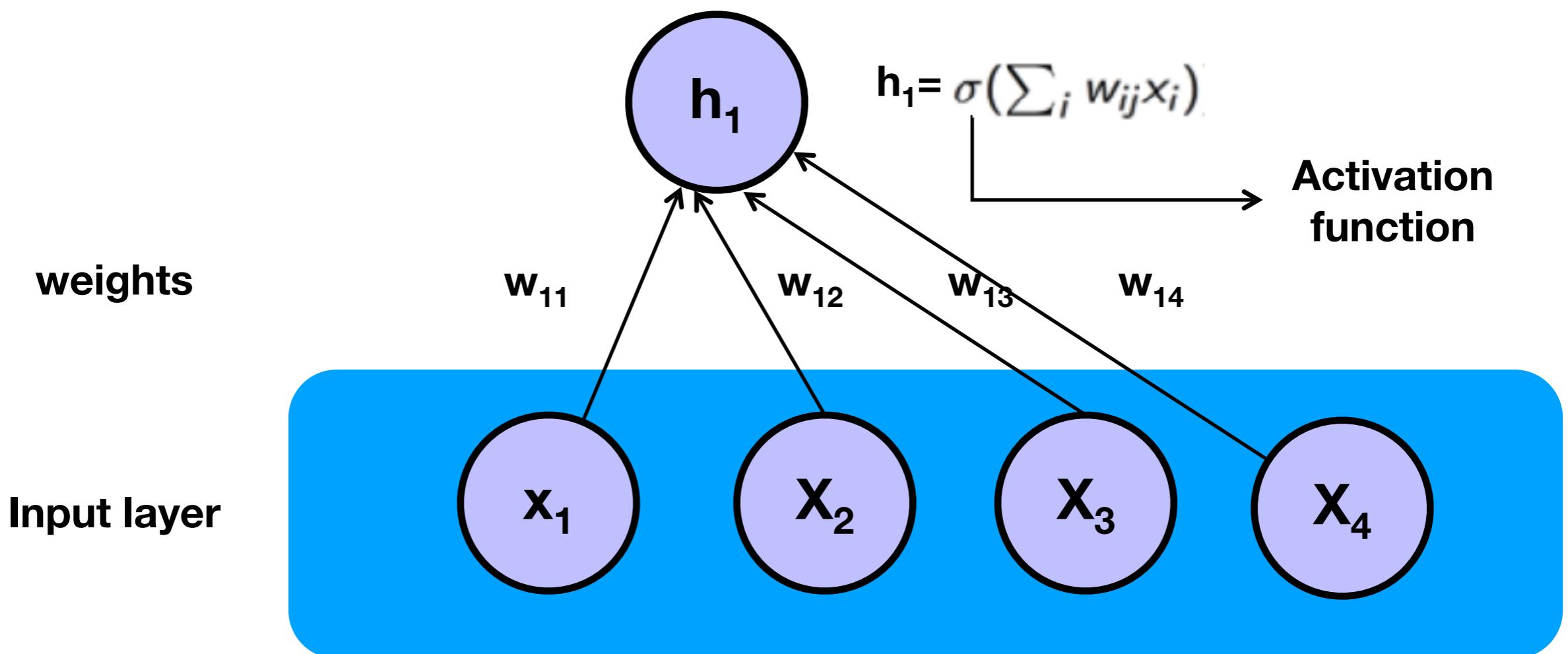
$$\hat{\theta} = \arg \min_{\theta} \mathcal{L} \left(\{x_i, y_i\}_{i \in \{1, \dots, N\}}; f_{\theta} \right)$$

Key questions:

- Which parameterisation for model f ?
- Which loss function ?

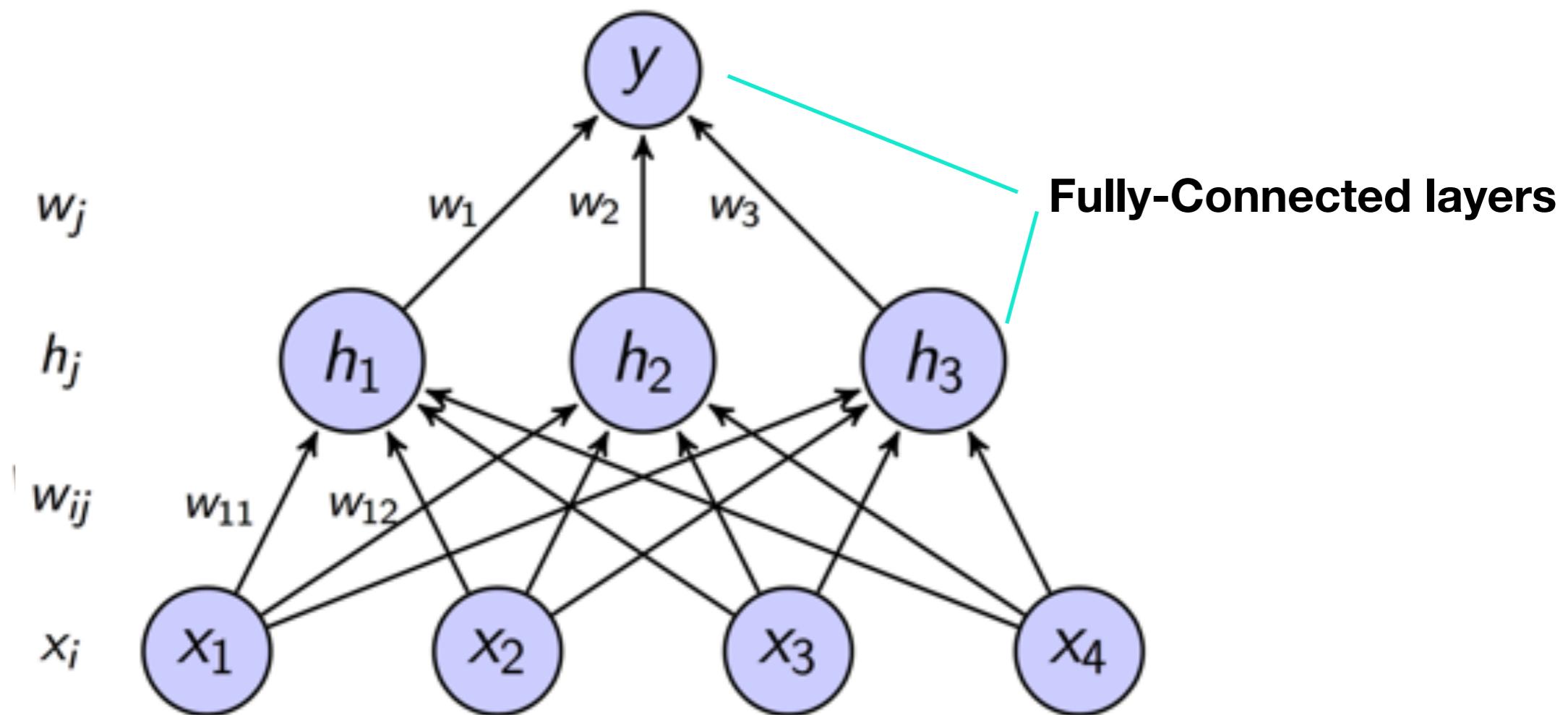
NEURAL NETWORKS: KEY PRINCIPLES & BUILDING BLOCKS

The artificial neuron



$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij}x_i))$$

Multilayer Perceptron (MLP)



$$f(x) = \sigma\left(\sum_j w_j \cdot h_j\right) = \sigma\left(\sum_j w_j \cdot \sigma\left(\sum_i w_{ij} x_i\right)\right)$$

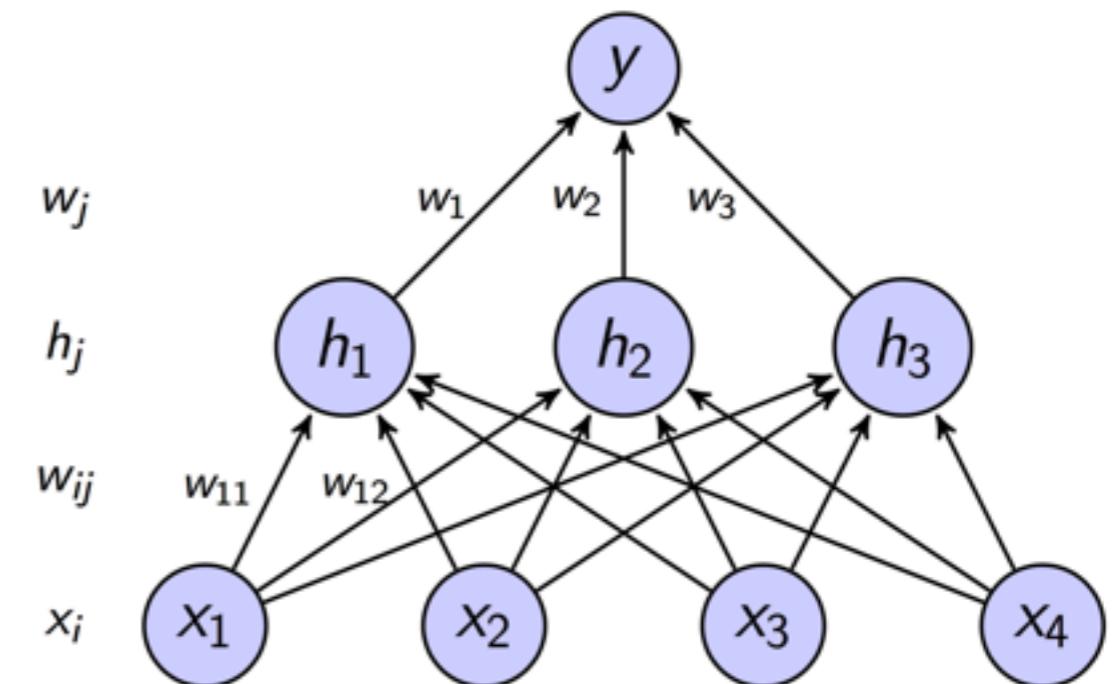
Multilayer Perceptron (MLP)

- **Neuron:** basic unit, inspired from neuroscience

- **Feedforward:** no feedback connection

- **Network:** composition of functions

- **Parametric model:** $y=f(x_1, x_2, x_3, x_4, \Theta)$



- **Training:** estimation of model parameter to minimise some loss function

**Let's play with MLPs using
tensorflow playground**

<http://playground.tensorflow.org/>

Neural networks: composition idea

- Approximation through the composition of (simple) elementary functions:

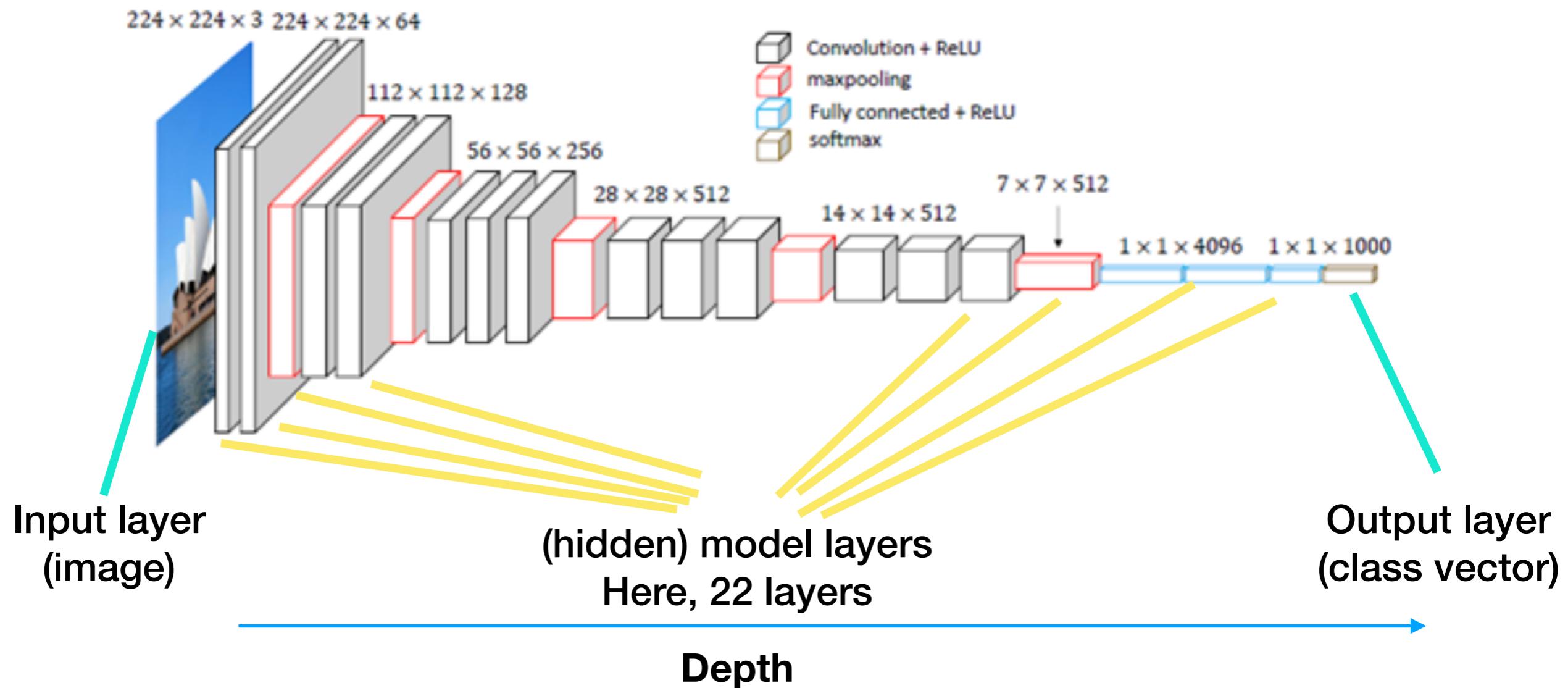
$$f_{\theta}(x) = f_{\theta_N} \circ \dots \circ f_{\theta_2} \circ f_{\theta_1}(x)$$

- Key features:
 - Any continuous function can be approximated as the composition of elementary functions
 - Analytical/exact computation of the derivative of f with respect to parameters and input variables
 - Direct exploitation of gradient-based optimisation schemes

**What are deep learning
models ?**

Basics of DL models

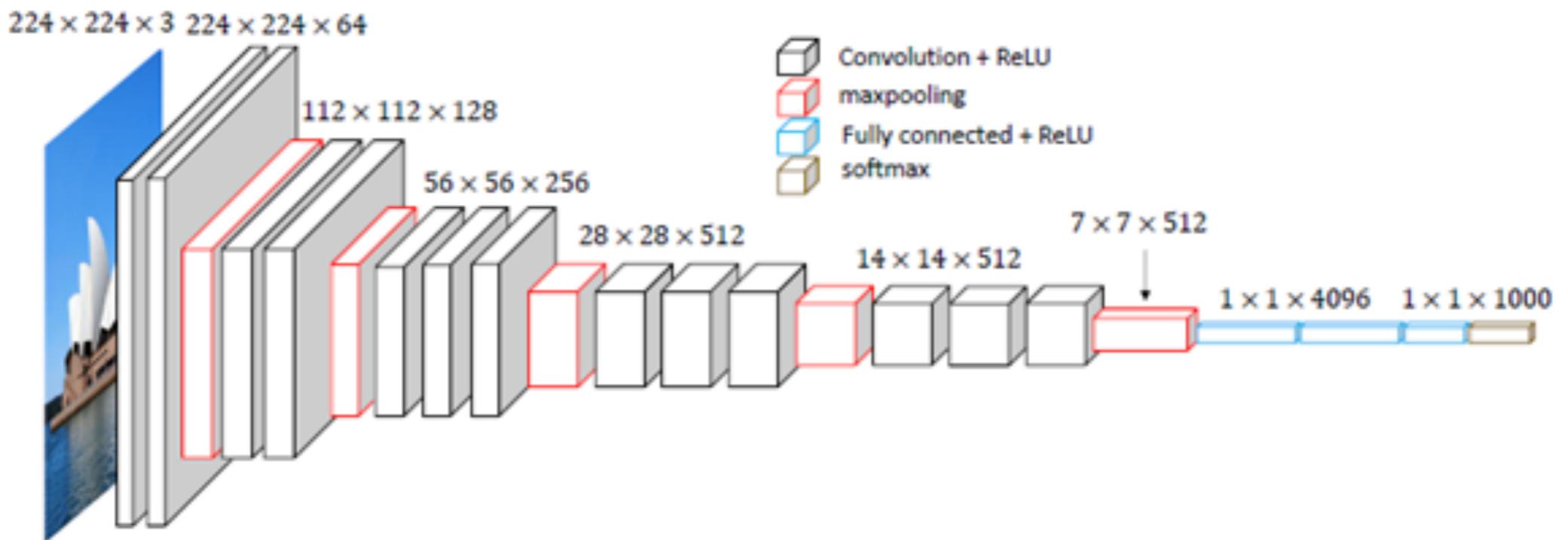
DL models are (in general) feedforward models. VGG16 as an illustration



The more layers, the deeper..... Some models may have up to several hundreds to thousands of layers.

Basics of DL models

DL models are (in general) feedforward models. VGG16 as an illustration



Elementary
components

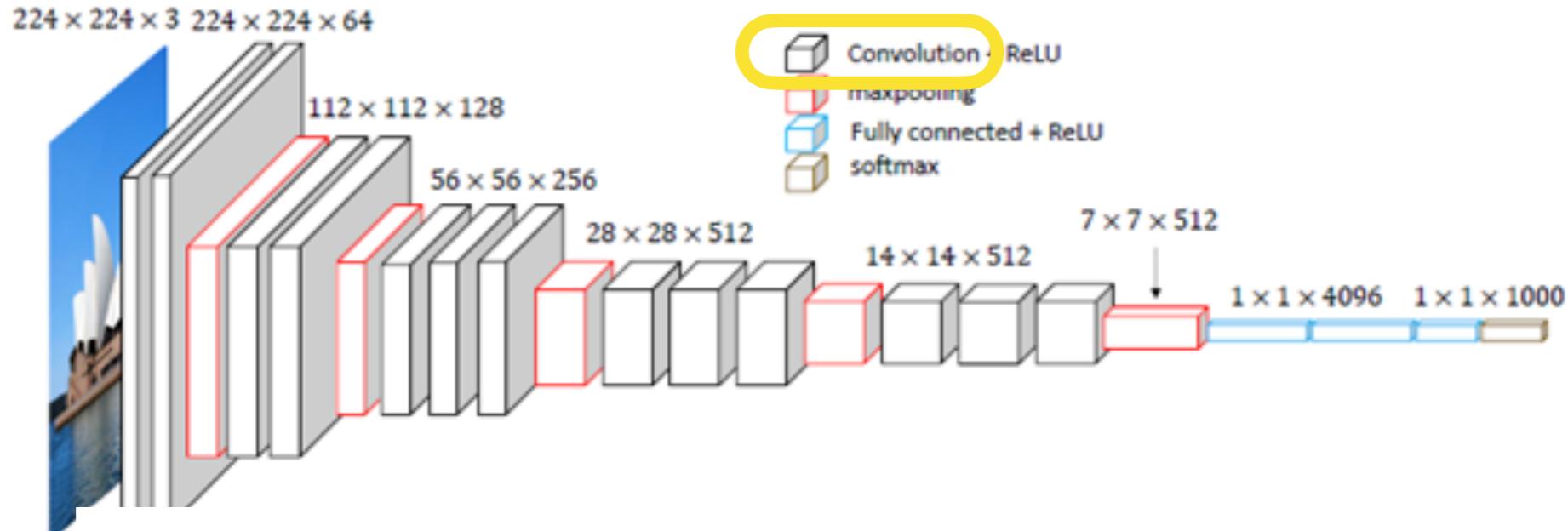
Convolution layers

Activation layers

Pooling layers

Dense layers

Basics of DL models



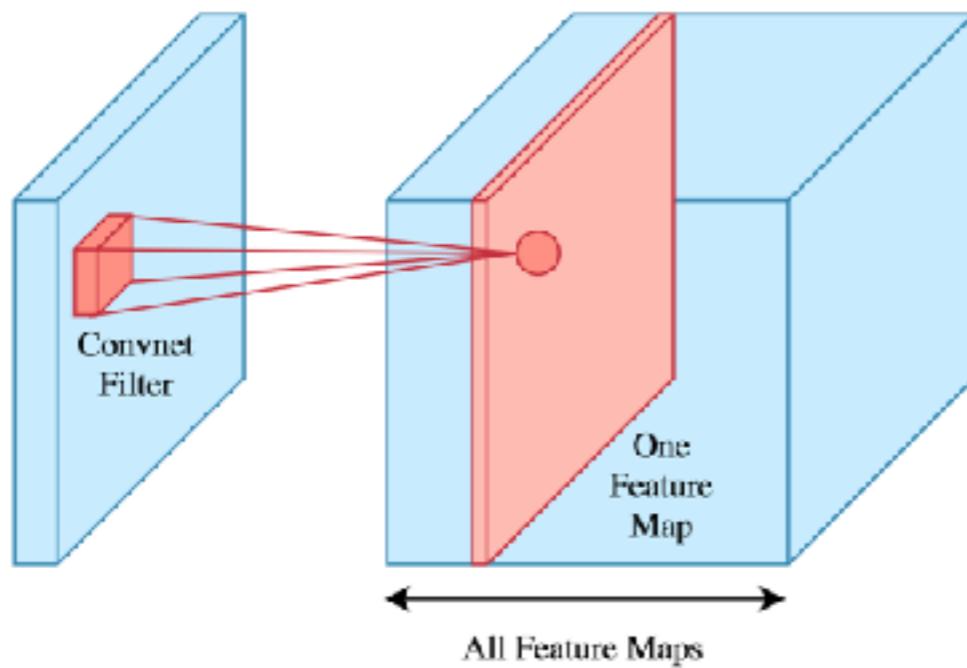
Elementary components

Convolution layers

Activation layers

Pooling layers

Dense layers

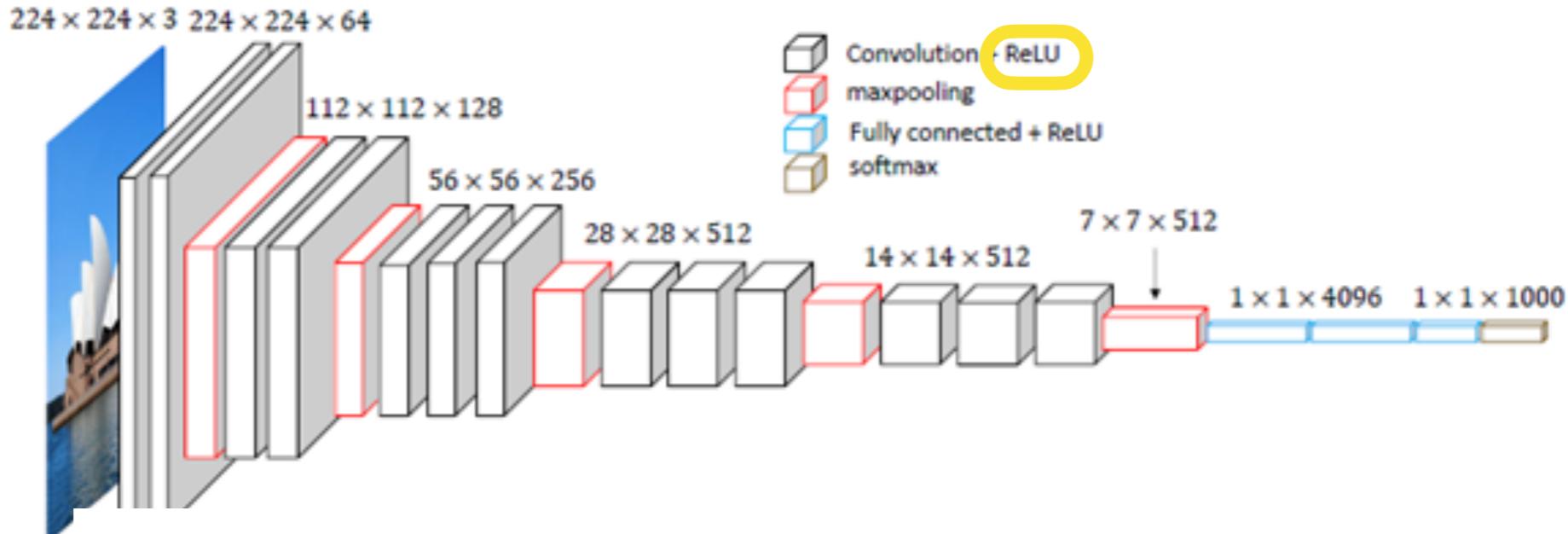


Number of parameters:
Filter size x Number of filters

e.g. $3 \times 3 \times K \times N_{\text{filt}}$

Independent on the sizes of the input
and output layer

Basics of DL models



Elementary components

Convolution layers

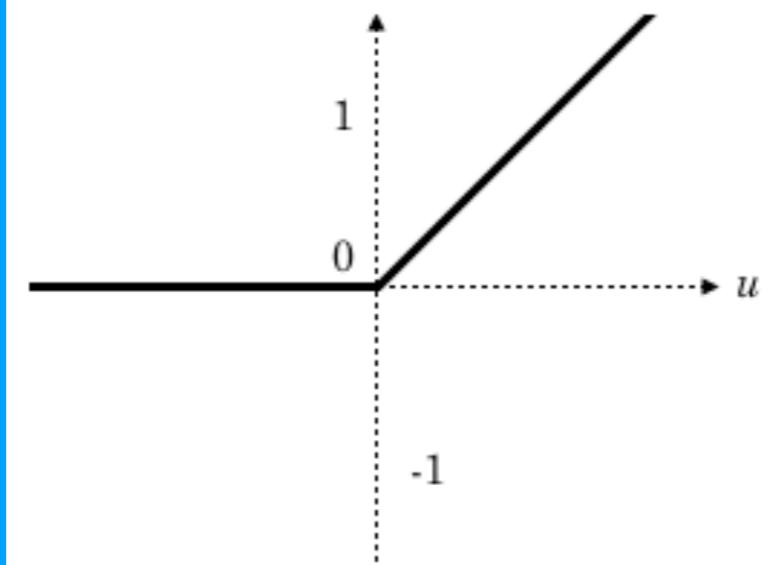
Activation layers

Pooling layers

Dense layers

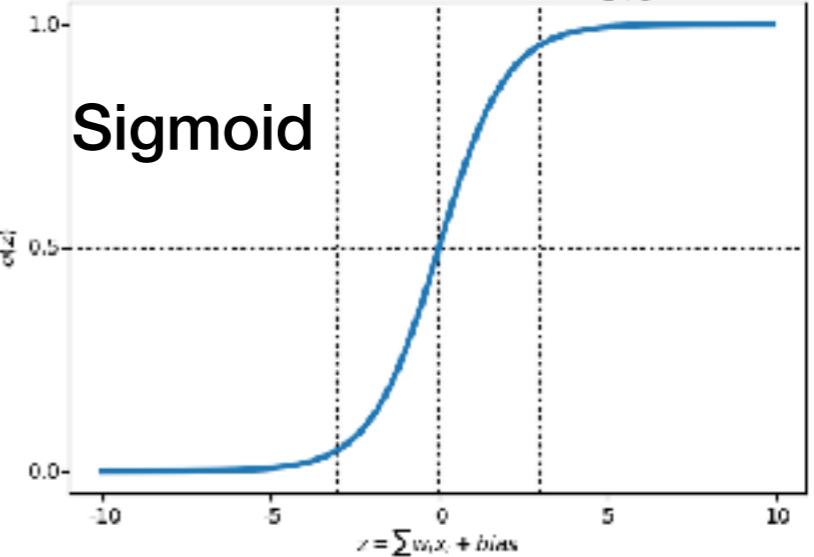
ReLU
(Rectified
Linear Unit)

$$f(u) = \max(0, u)$$

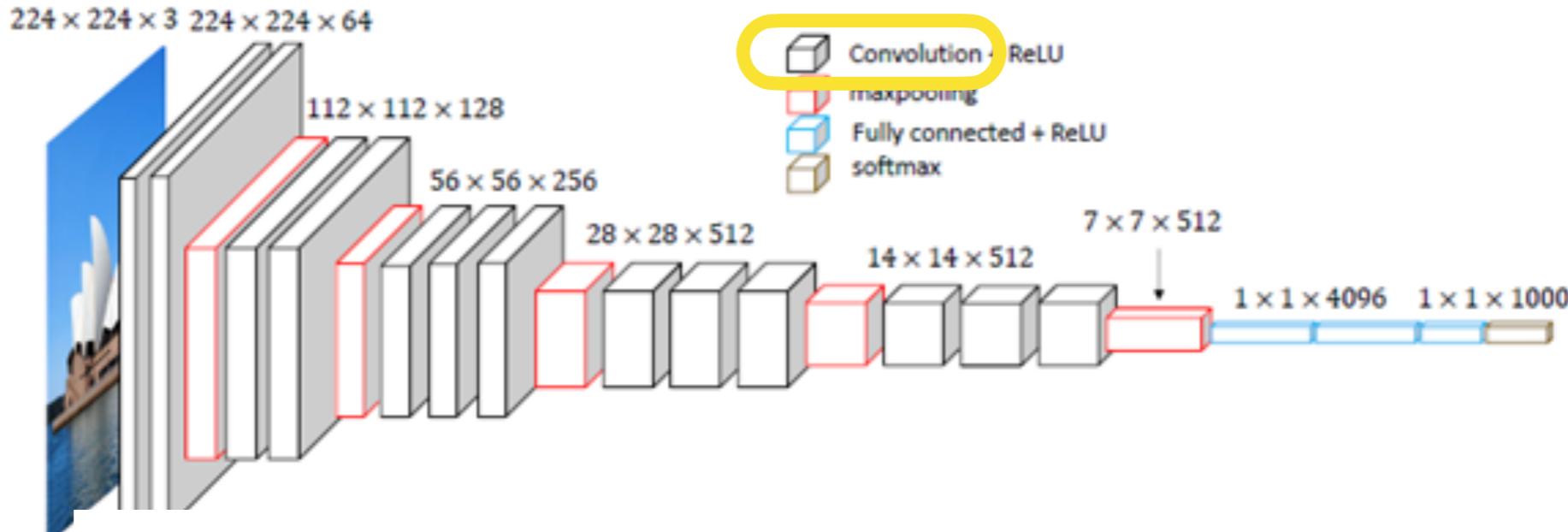


Sigmoid Function $\sigma(z) = \frac{1}{1+e^{-z}}$

Sigmoid



Basics of DL models



Elementary components

Convolution layers

Activation layers

Pooling layers

Dense layers

Number of parameters:

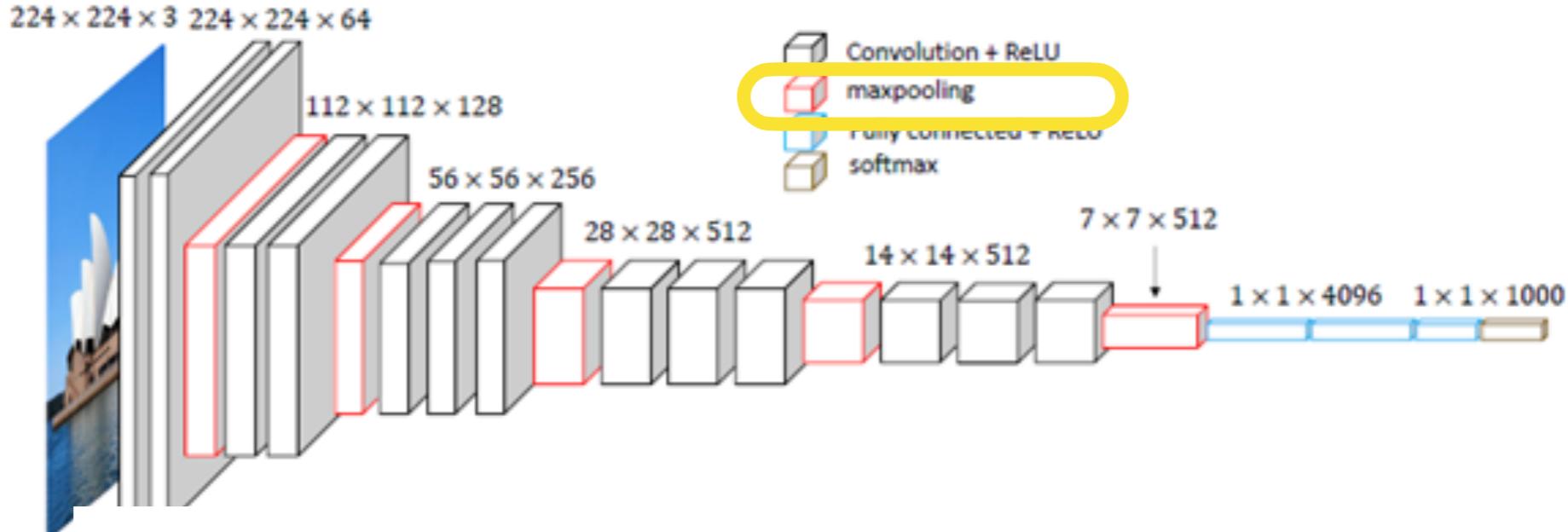
Keras function

```
model.add(Conv2D(32, kernel_size=(3, 3),  
                activation='relu',  
                input_shape=input_shape))
```

put

All Feature Maps

Basics of DL models



Elementary components

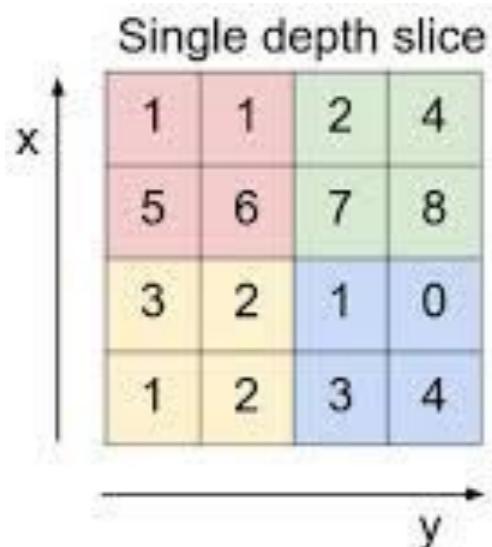
Convolution layers

Activation layers

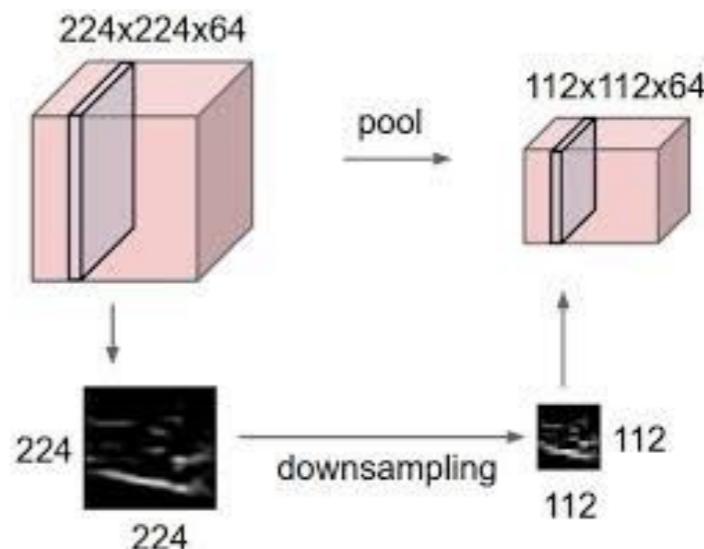
Pooling layers

Dense layers

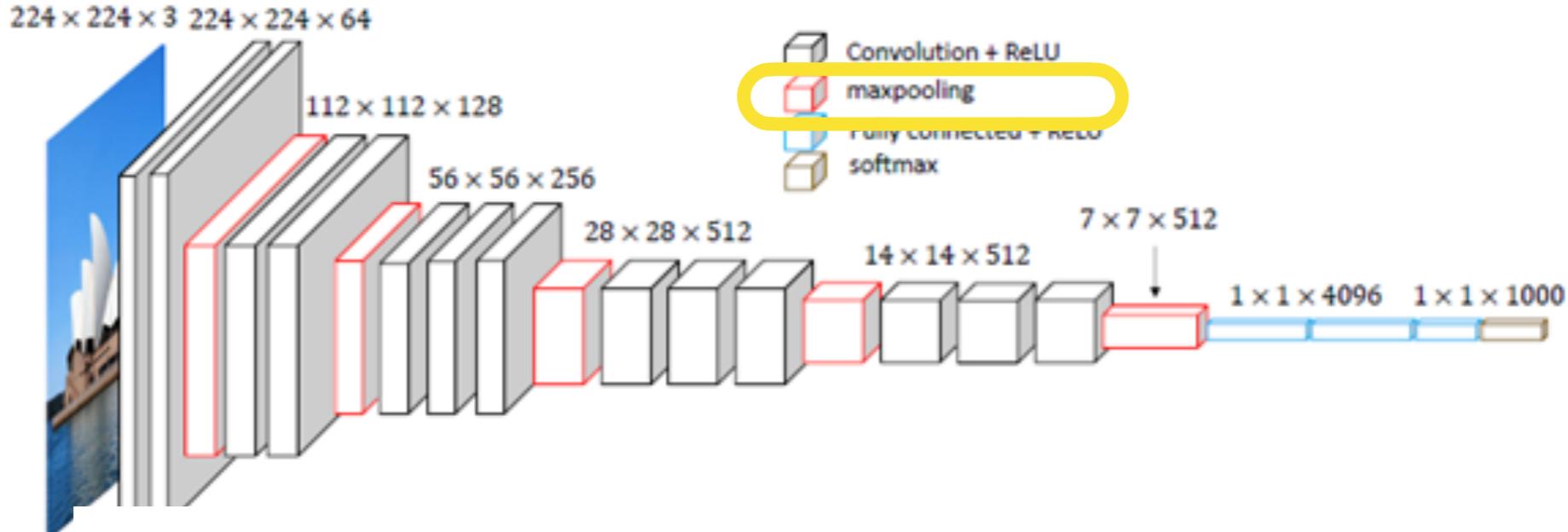
An example of max-pooling operator



Pooling downsamples the input layer



Basics of DL models



Elementary components

Convolution layers

Activation layers

Pooling layers

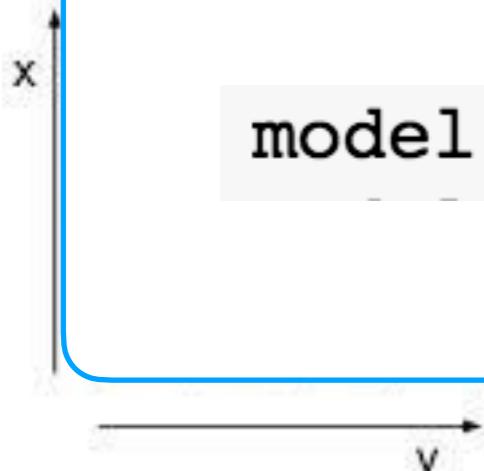
Dense layers

An example of max-pooling operator

Pooling downsamples the input layer

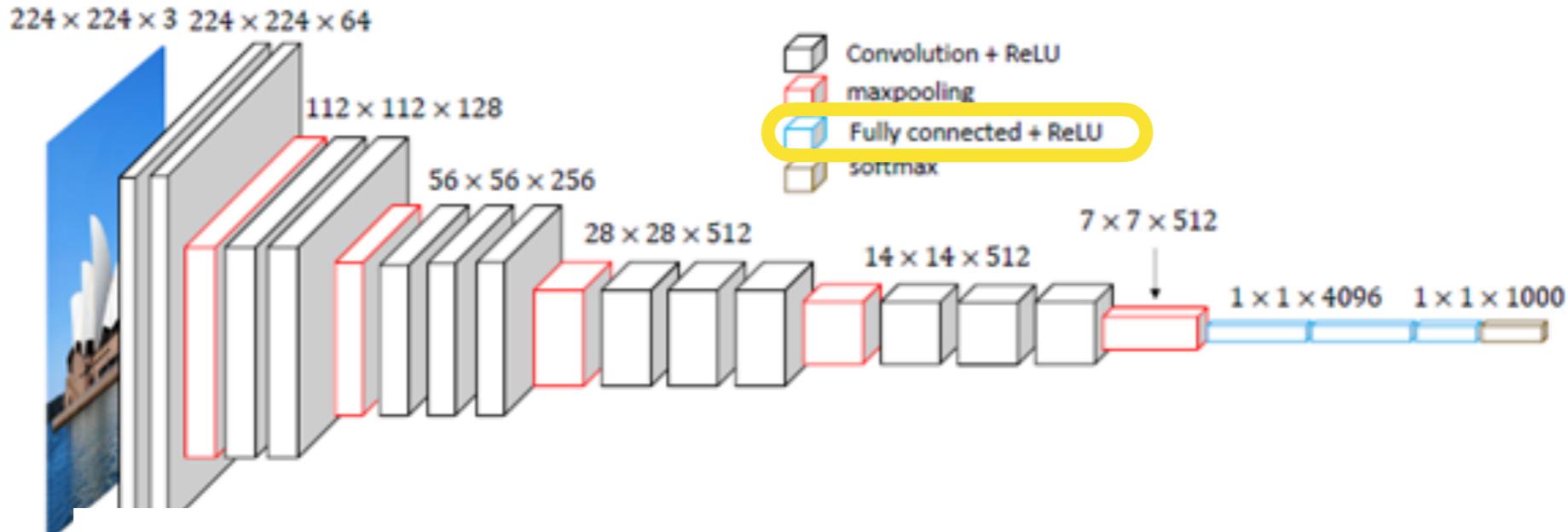
Keras function

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```



224 downsampling 112
224 112

Basics of DL models



Elementary components

Convolution layers

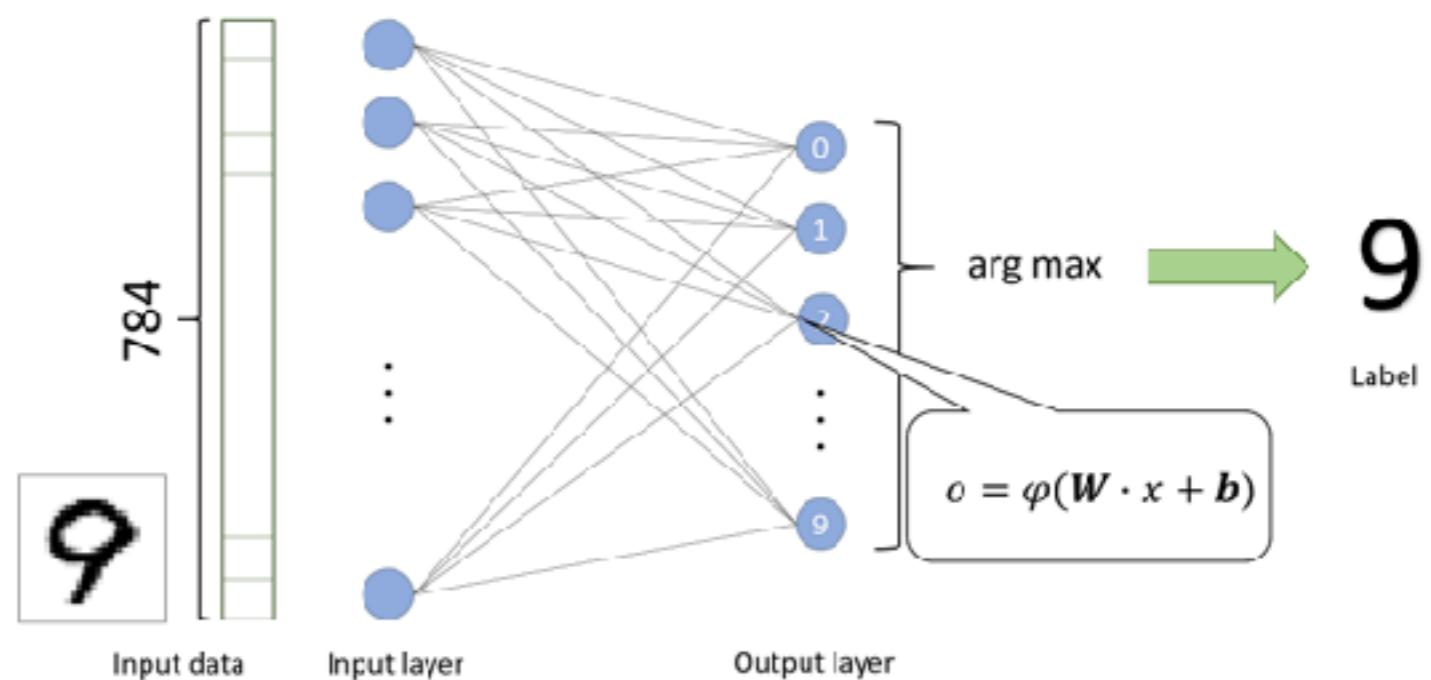
Activation layers

Pooling layers

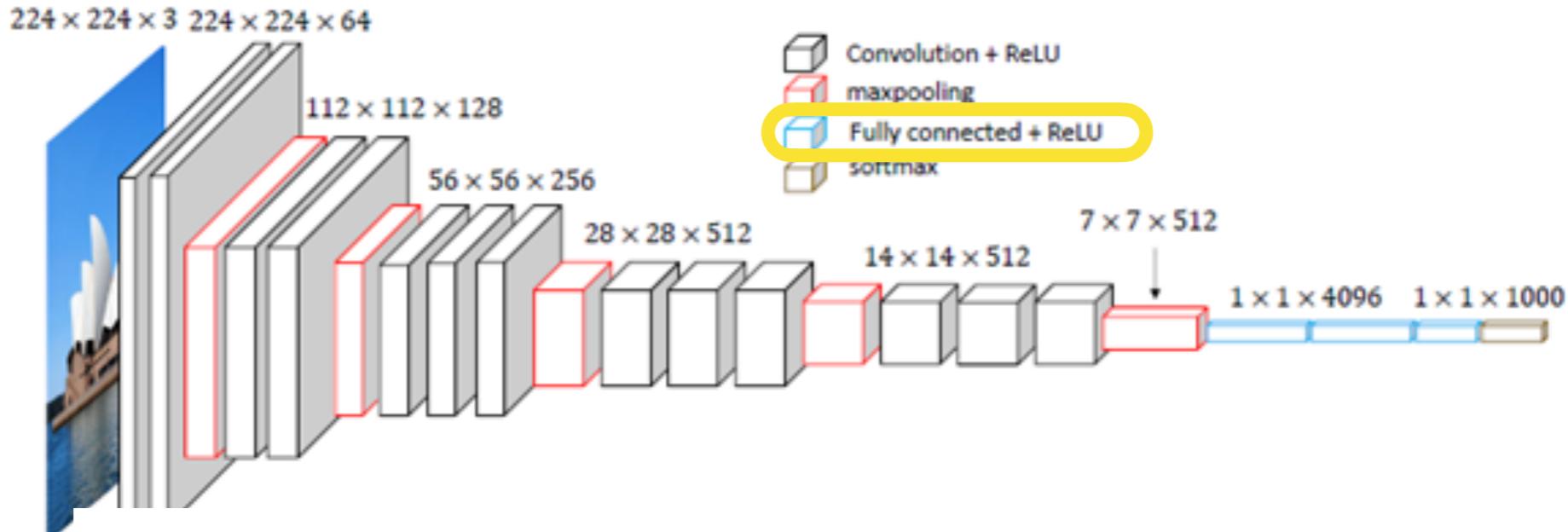
Dense layers

Dense layers
or
Fully-connected (FC) layer

as in a classic MLP



Basics of DL models



Elementary components

Convolution layers

Activation layers

Pooling layers

Dense layers



Keras function

```
model.add(Dense(128, activation='relu'))  
model.add(Dense(num_classes, activation='softmax'))
```



Input data

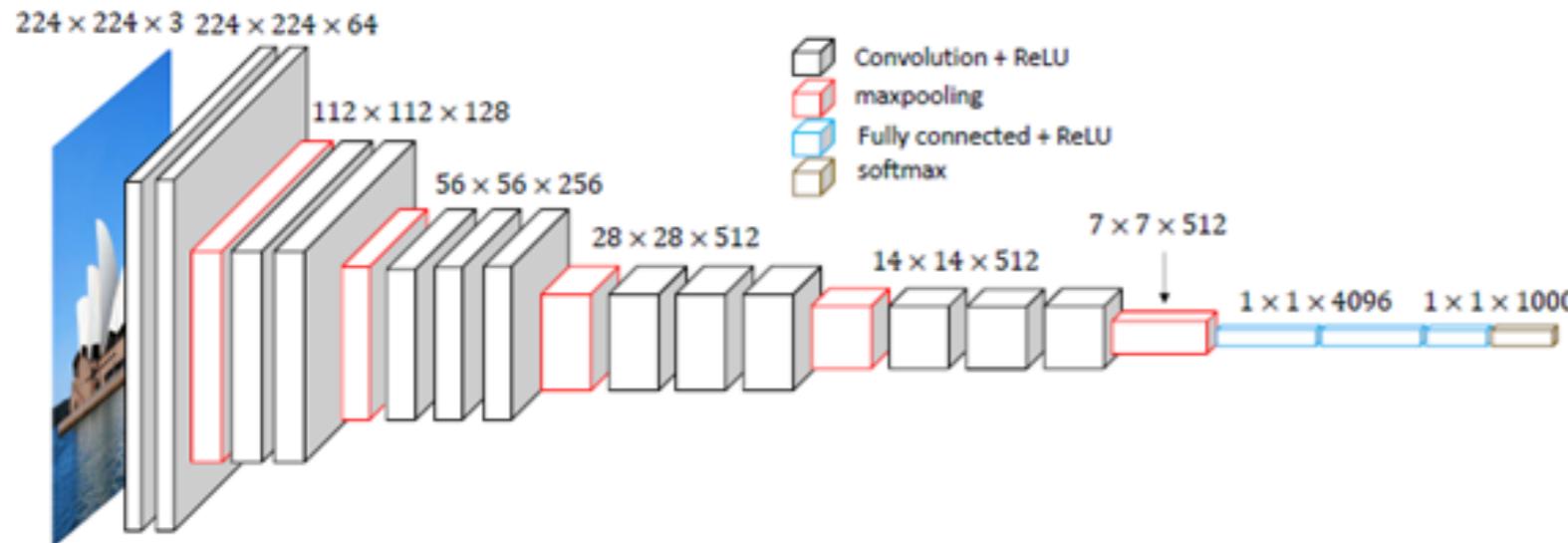


Input layer



Output layer

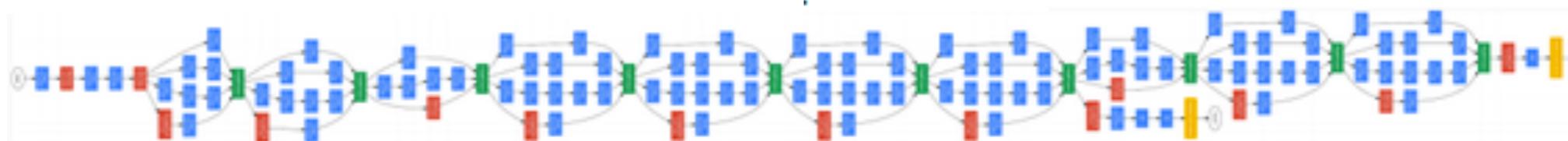
Examples of DL models for object recognition



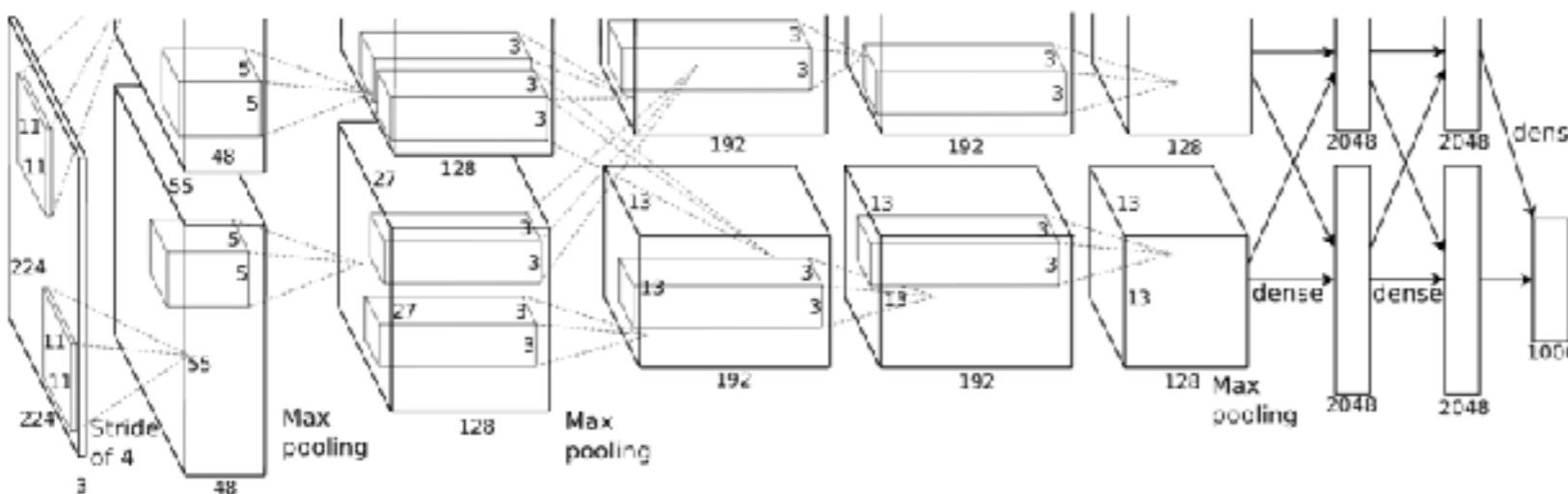
VGG16
(<100M of parameters)

Google inception

(5M of parameters)



Convolution
Pooling
Softmax
Other

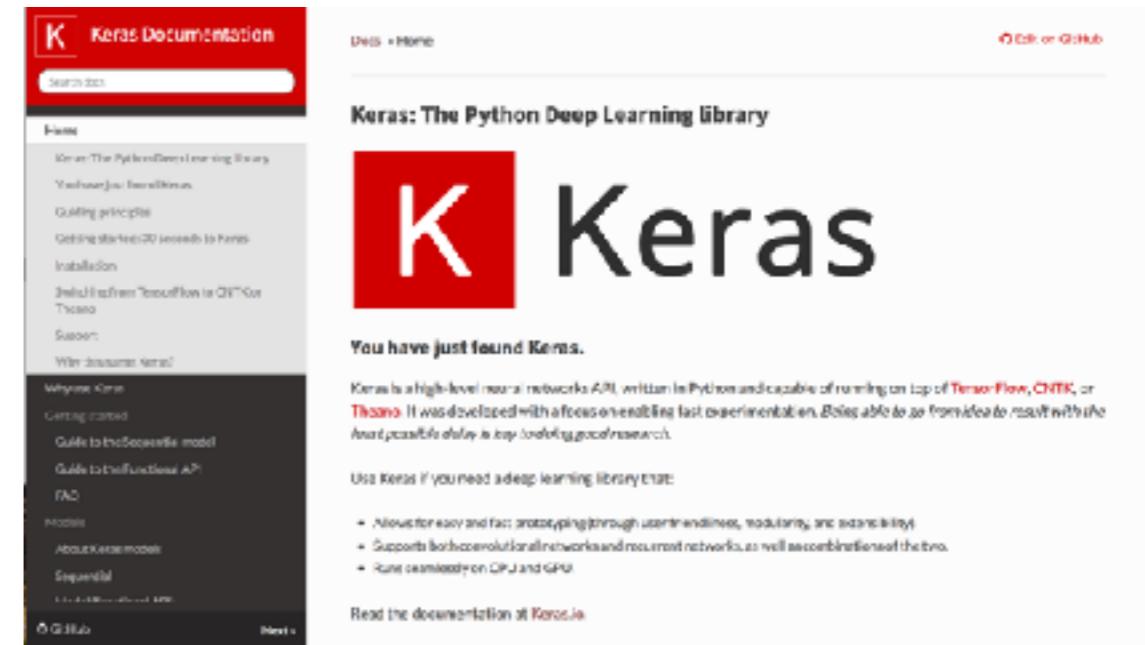


AlexNet
(60M of parameters)

How to implement Neural Nets with Keras ?

Using neural nets with Keras

- What is Keras ?
 - A high-level library for DL using Tensorflow or Theano as low-level DL library
- Not the only Python DL library



	PyTorch	Tensorflow	Keras
+	Mid-level, modelling flexibility, supported by Facebook AI Research	Low-level, computational efficiency, developed by Google	High-level, simple-to-use, benefits from using Tensorflow as backend
-	in-between in terms of use simplicity and computational complexity	relatively complex to develop with	Lower modelling flexibility

Load Python notebook

LabSession_MLP_MNIST.ipynb

Multilayer Perceptron (MLP)

- A simple regression example (MNIST case-study)

- Goal: predict a digit from an image



- Formulation: non-linear regression using a softmax model

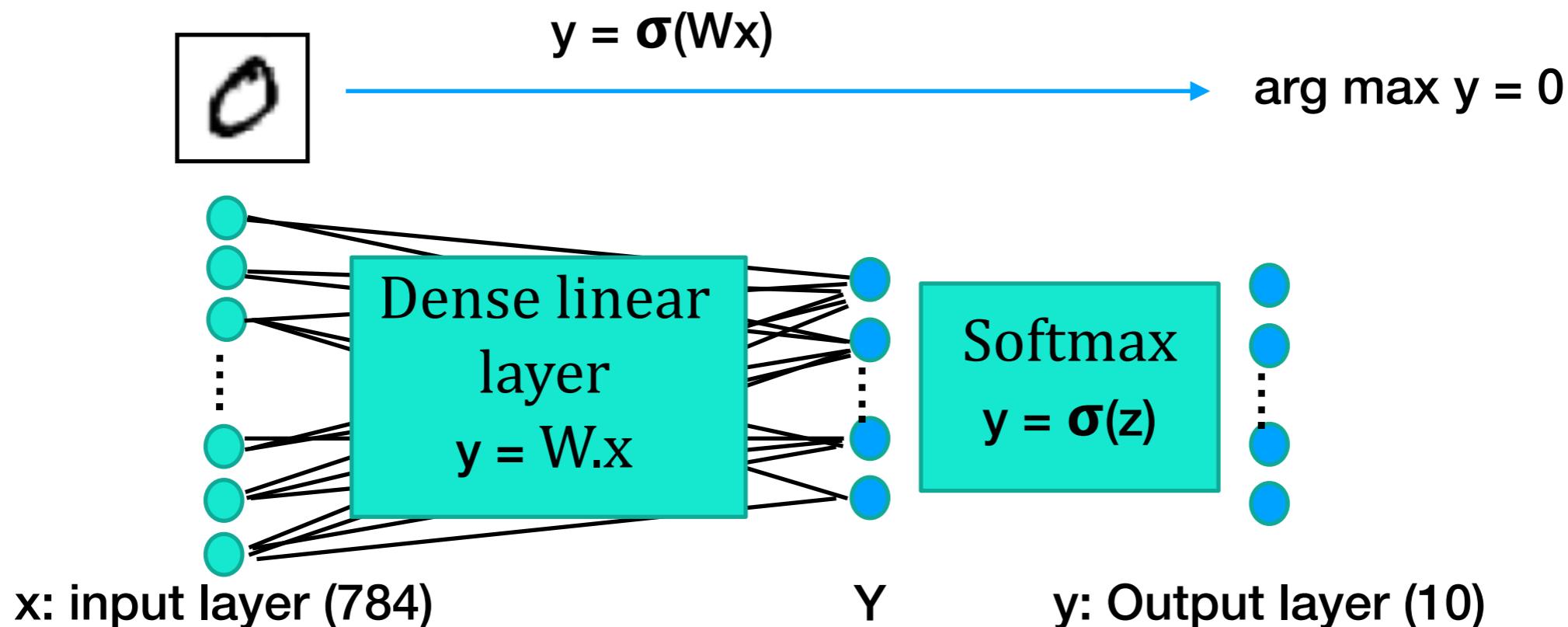
$$\begin{array}{ccc} \text{} & \xrightarrow{\text{y} = \sigma(\mathbf{W} \cdot \mathbf{x})} & 0 \\ \text{x (28x28 image)} & & \text{y} \end{array} \quad \text{with } \sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K.$$

A diagram illustrating the formulation of softmax regression. On the left, a handwritten digit '0' is shown inside a box, labeled 'x (28x28 image)'. An arrow labeled $y = \sigma(\mathbf{W} \cdot \mathbf{x})$ points from the digit to the right, where the variable 'y' is shown. To the right of the arrow is the formula for the softmax function, which maps a vector \mathbf{z} to a probability distribution over K classes, indexed by j .

- Softmax regression as a one-layer MLP ?

Using neural nets with Keras

- Back to Softmax regression as a one-layer MLP



Building a model with Keras

```
from keras.models import Sequential  
from keras.layers import Dense  
  
# y = softmax (Wx+b)  
model = Sequential()  
model.add(Dense(num_classes, activation='softmax', input_shape=(784,)))  
  
model.summary()
```

Declare a new empty model

Add a dense layer

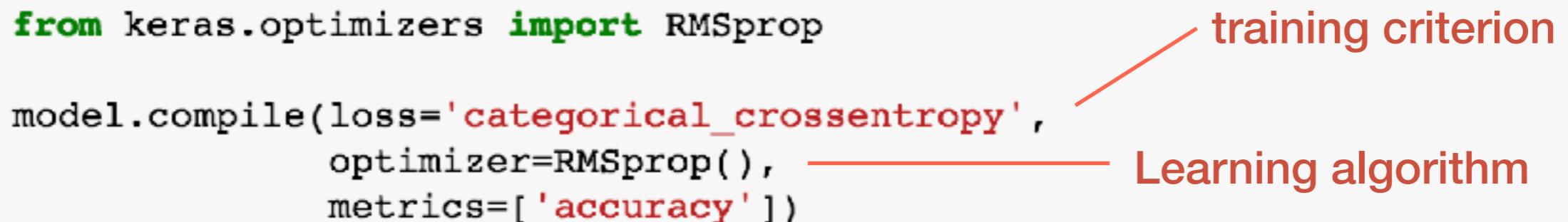
Using neural nets with Keras

- Back to Softmax regression as a one-layer MLP

Using a model with Keras

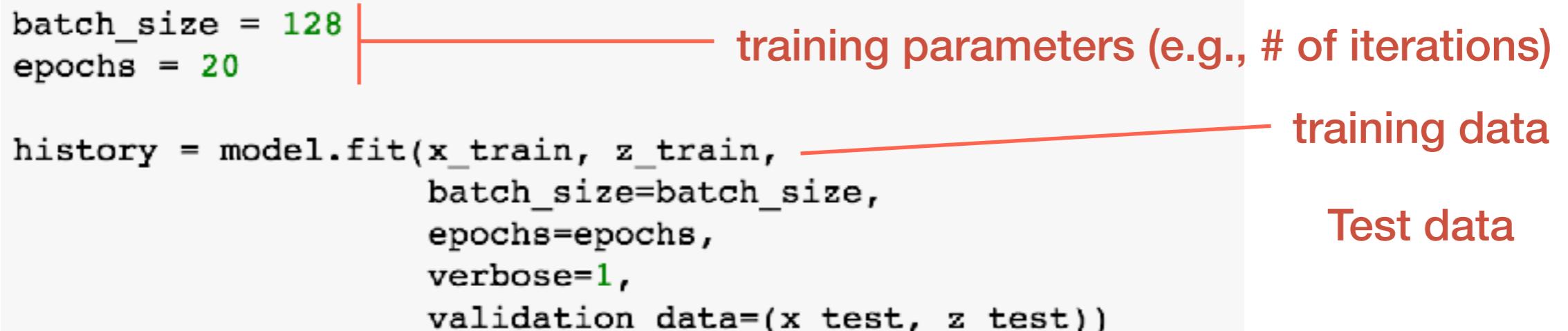
1. Model “compilation”

```
from keras.optimizers import RMSprop  
  
model.compile(loss='categorical_crossentropy',  
              optimizer=RMSprop(),  
              metrics=['accuracy'])
```



2. Model training from data

```
batch_size = 128  
epochs = 20  
  
history = model.fit(x_train, z_train,  
                     batch_size=batch_size,  
                     epochs=epochs,  
                     verbose=1,  
                     validation_data=(x_test, z_test))
```



3. Application of a trained model to new data

```
z_pred = model.predict(x_test)
```

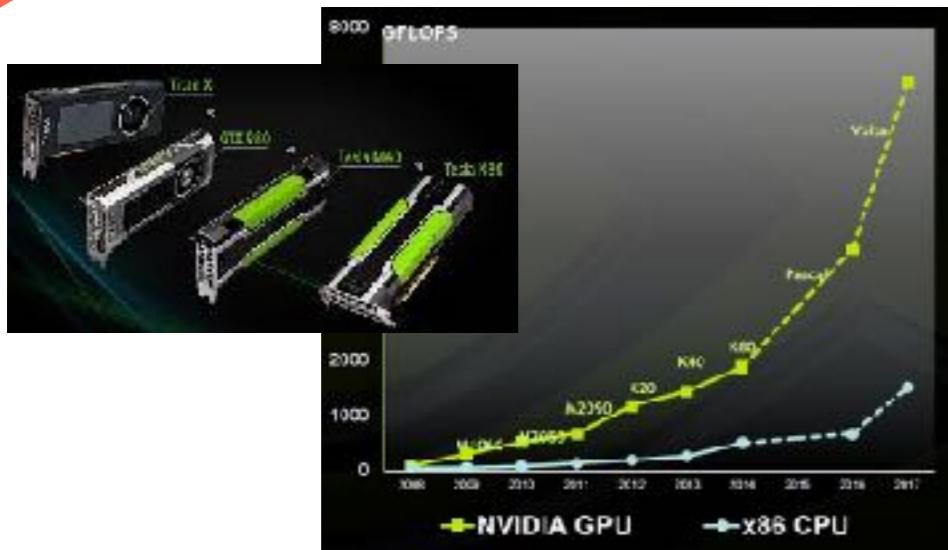


Load Python notebook

LabSession_CNN_MNIST.ipynb

**Why are Deep Learning
models so successful?**

Key reasons for DL emergence



High-performance computing (GPU)



Large annotated dataset (> 1M)

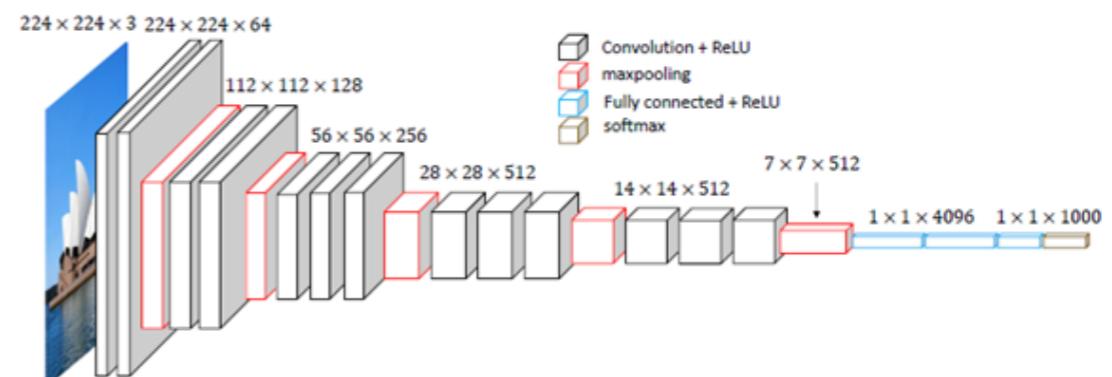


K Keras

Caffe

PYTORCH

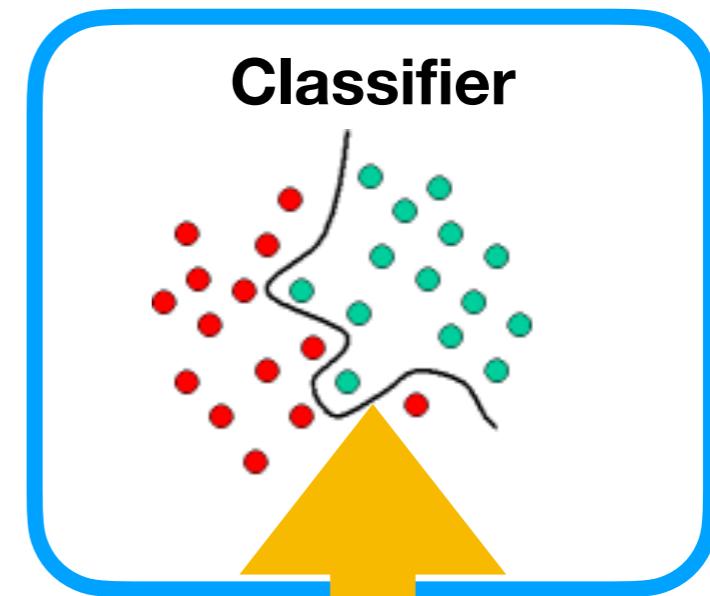
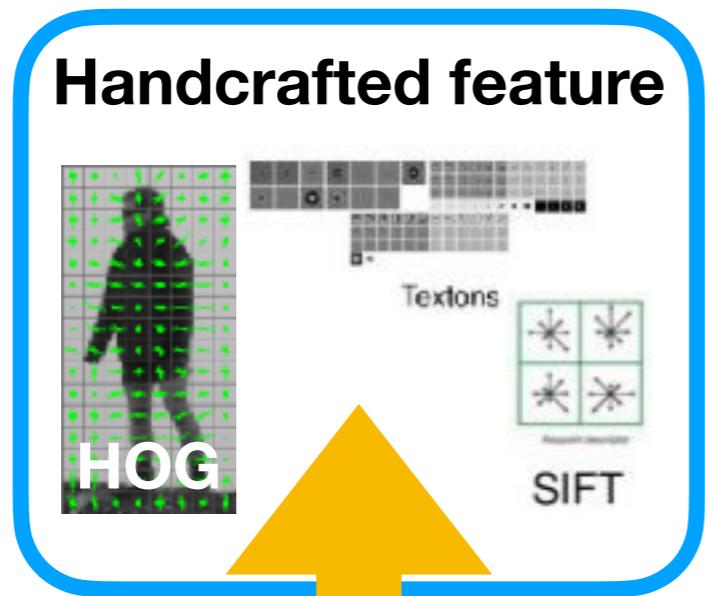
Efficient & easy-to-use libraries



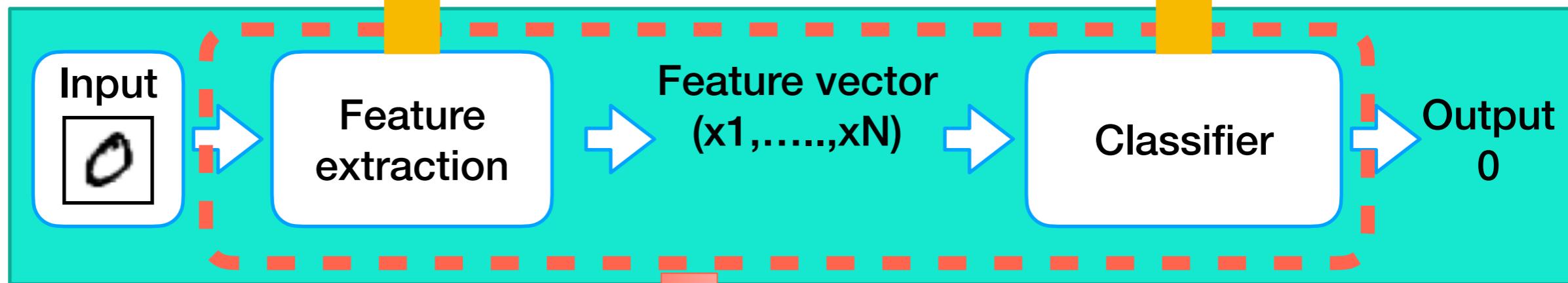
End-to-end learning

Shallow vs. Deep models

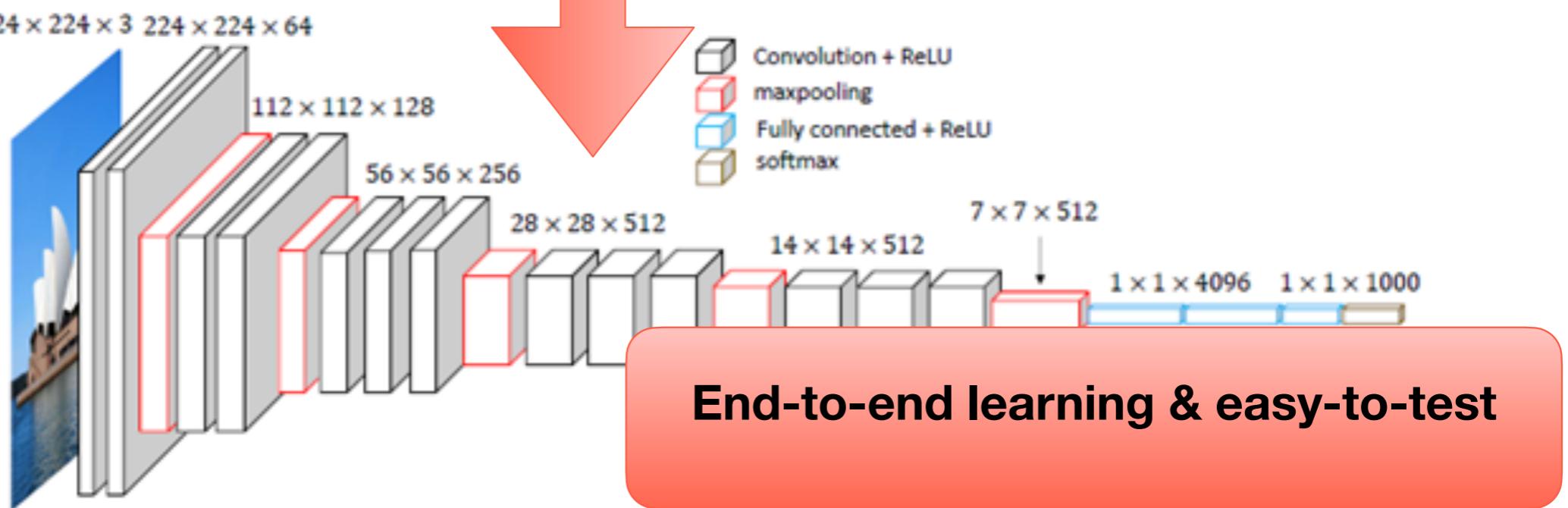
Shallow Learning



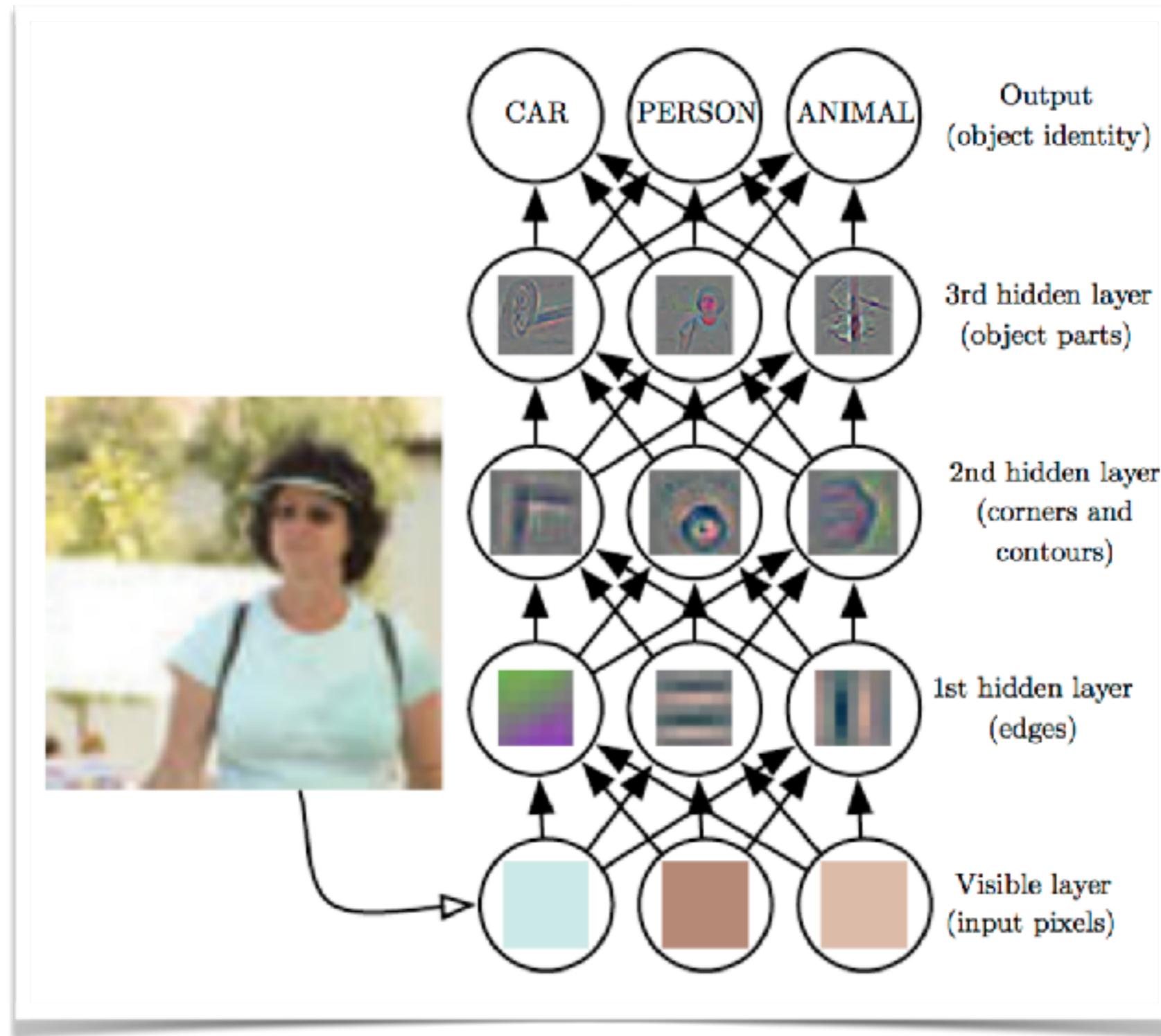
Classic Workflow



Deep Learning



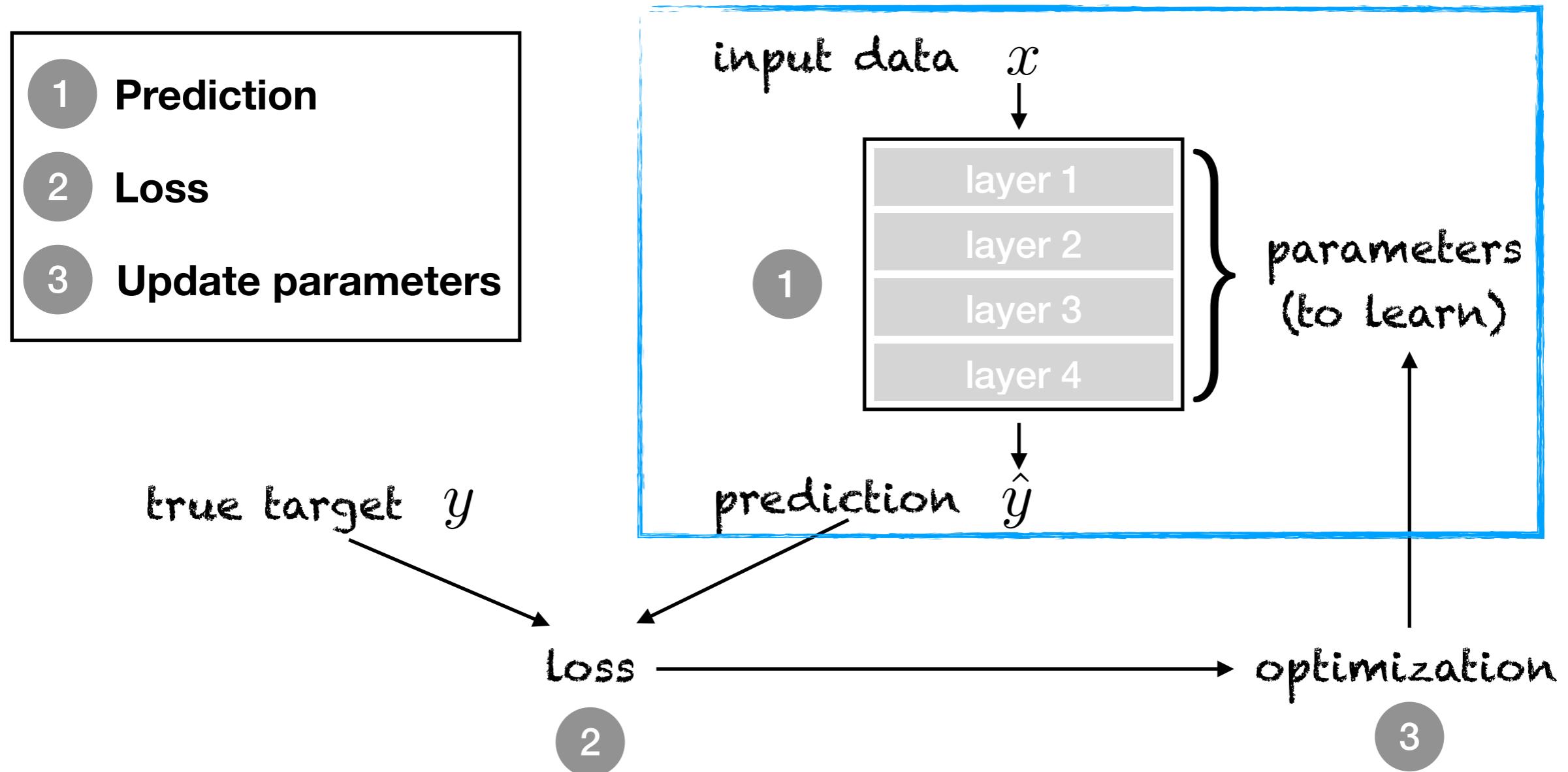
Feature extraction & Deep learning



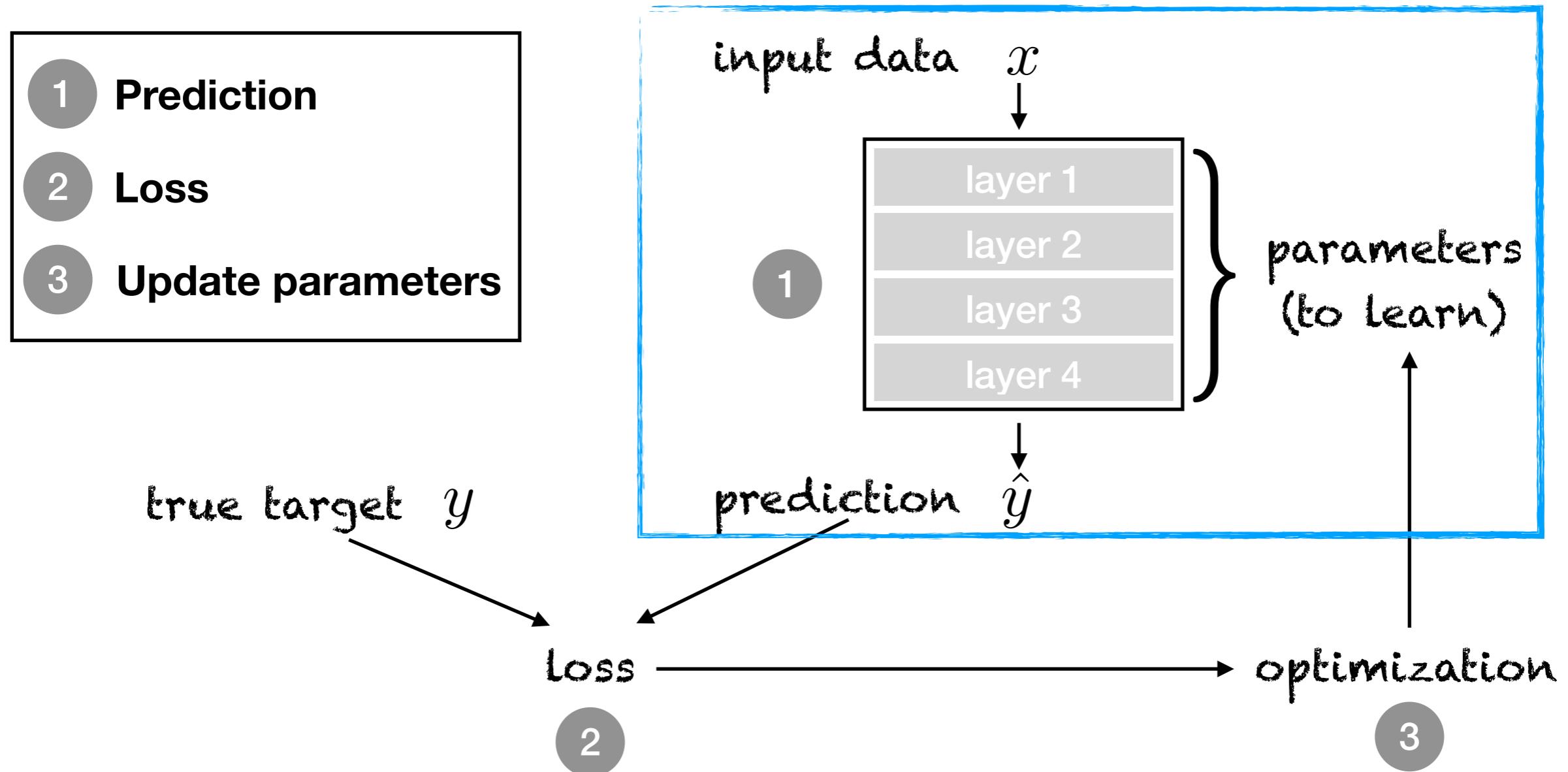
Summary

Tips for using DL models

Overview



Overview



Implementation vs theory

7	9	0	2	3	4	7	2	9	0	1	2	5	0	5	6	7	8	4
1	1	4	5	6	3	5	3	8	0	0	5	4	7	5	3	0	7	
3	5	6	2	7	1	3	7	6	2	9	3	7	0	6	7	4	1	2
7	5	1	5	1	2	8	0	2	6	9	7	3	0	7	2	5		
3	9	1	5	4	5	6	7	3	7	0	0	5	3	9	5	4	7	0
1	2	3	4	5	6	7	2	0	1	0	5	5	1	7	0	8	1	9
7	1	4	2	7	5	2	6	5	2	3	3	7	1	4	8	5	6	8
7	0	0	2	2	7	1	3	0	8	7	2	7	4	2	0	3	4	
8	7	1	5	9	2	3	9	9	4	0	5	3	2	6	3	8	7	
3	5	0	1	3	6	4	6	7	8	9	0	1	2	8	4	5	6	7
8	9	0	1	5	6	2	5	6	7	8	9	5	4	2	6	3	5	6
9	2	8	5	4	3	9	3	5	0	1	8	0	5	6	0	1	0	
4	2	6	5	3	5	4	1	6	3	0	6	3	0	5	6	0	1	
4	5	1	2	3	8	5	4	2	0	5	7	4	1	6	8	4		
7	5	1	2	6	7	1	3	6	0	6	9	2	9	6	2	3	7	1
7	2	2	5	5	7	8	0	1	2	3	4	7	6	7	8	0	1	2
7	9	6	7	8	0	1	2	3	4	7	6	7	8	0	1	2	3	
2	4	8	5	2	7	0	9	7	5	2	4	3	7	0	3	5	1	9
8	9	5	2	1	4	3	8	0	6	4	8	4	2	5	7	2	5	9

1. DATA

```
from keras.models import Sequential
from keras.layers import Dense

#  $y = \text{softmax}(Wx+b)$ 
model = Sequential()
model.add(Dense(num_classes, activation='softmax', input_shape=(784,)))

model.summary()
```

3. NETWORK

```
from keras.optimizers import RMSprop

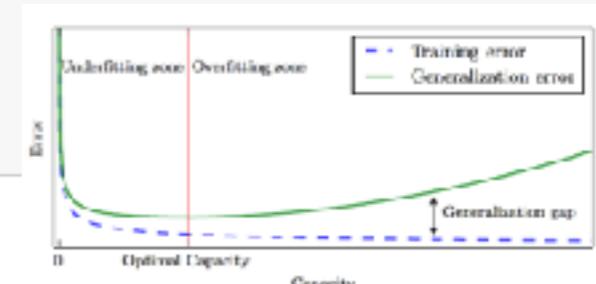
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])
```

2. LOSS FUNCTION

```
batch_size = 128
epochs = 20

history = model.fit(x_train, z_train,
                     batch_size=batch_size,
                     epochs=epochs,
                     verbose=1,
                     validation_data=(x_test, z_test))
```

4. LEARNING



Tips for using DL models

- 1. State the considered issue as a machine learning problem**
- 2. Build a (large-scale) groundtruthed dataset**
- 3. Design a DL architecture (combination of conv, pooling, dense layers)**
Possibility for using existing architectures (fine-tuning, transfer learning)
- 4. Buy or rent computational ressources (especially, GPU resources)**
- 5. Train the selected DL architecture**
Here comes the “tricky” part of the selection of the optimisation strategy

Training and Optimisation issues in Deep Learning

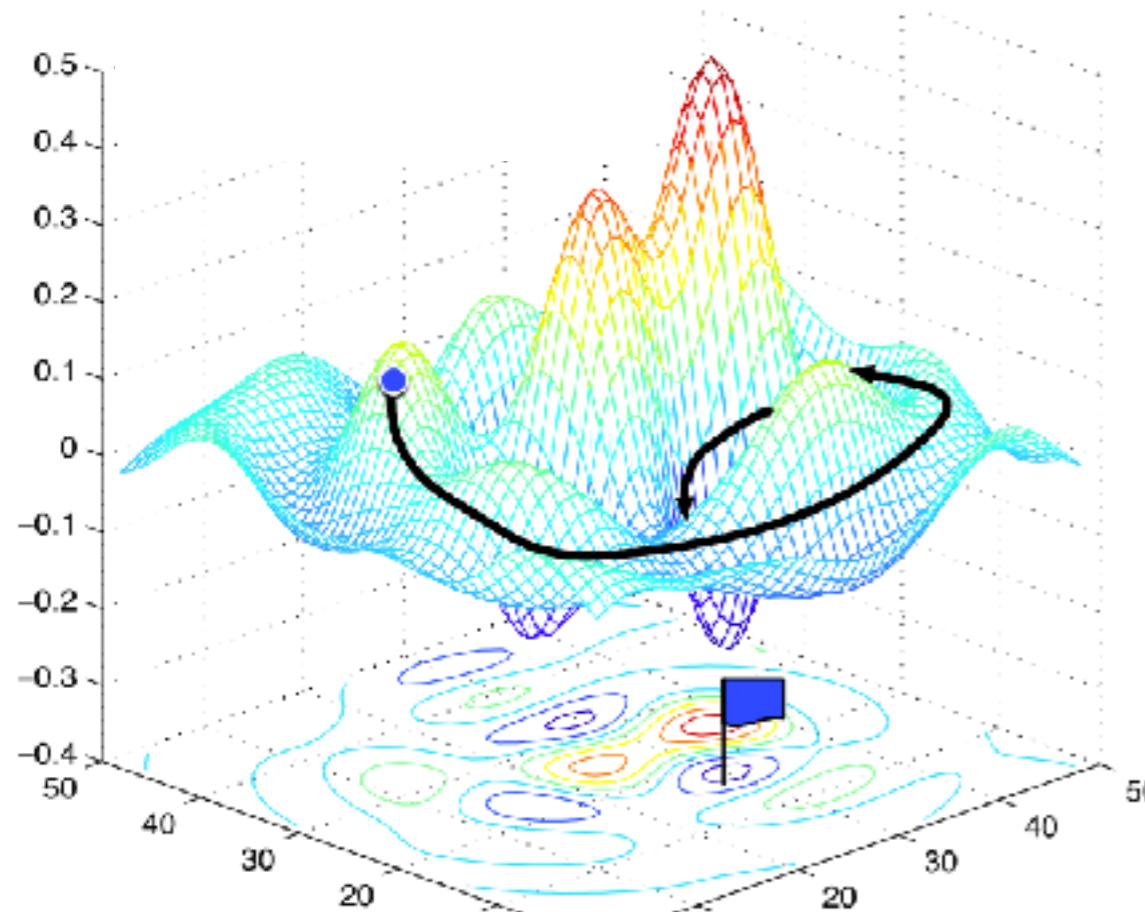
All you need is Data, GPUs

And a good optimisation algorithm

Training and Optimisation issues in Deep Learning

All you need is Data, GPUs

And a good optimisation algorithm



Non-convex optimisation using gradient-based algorithm

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)$$

Model parameters

Gradient of the training loss

A short tour of the NNs zoo

2 (at least) key aspects

NN architecture (parameterization of model f)
Training loss

Autoencoder and dimensionaly reduction

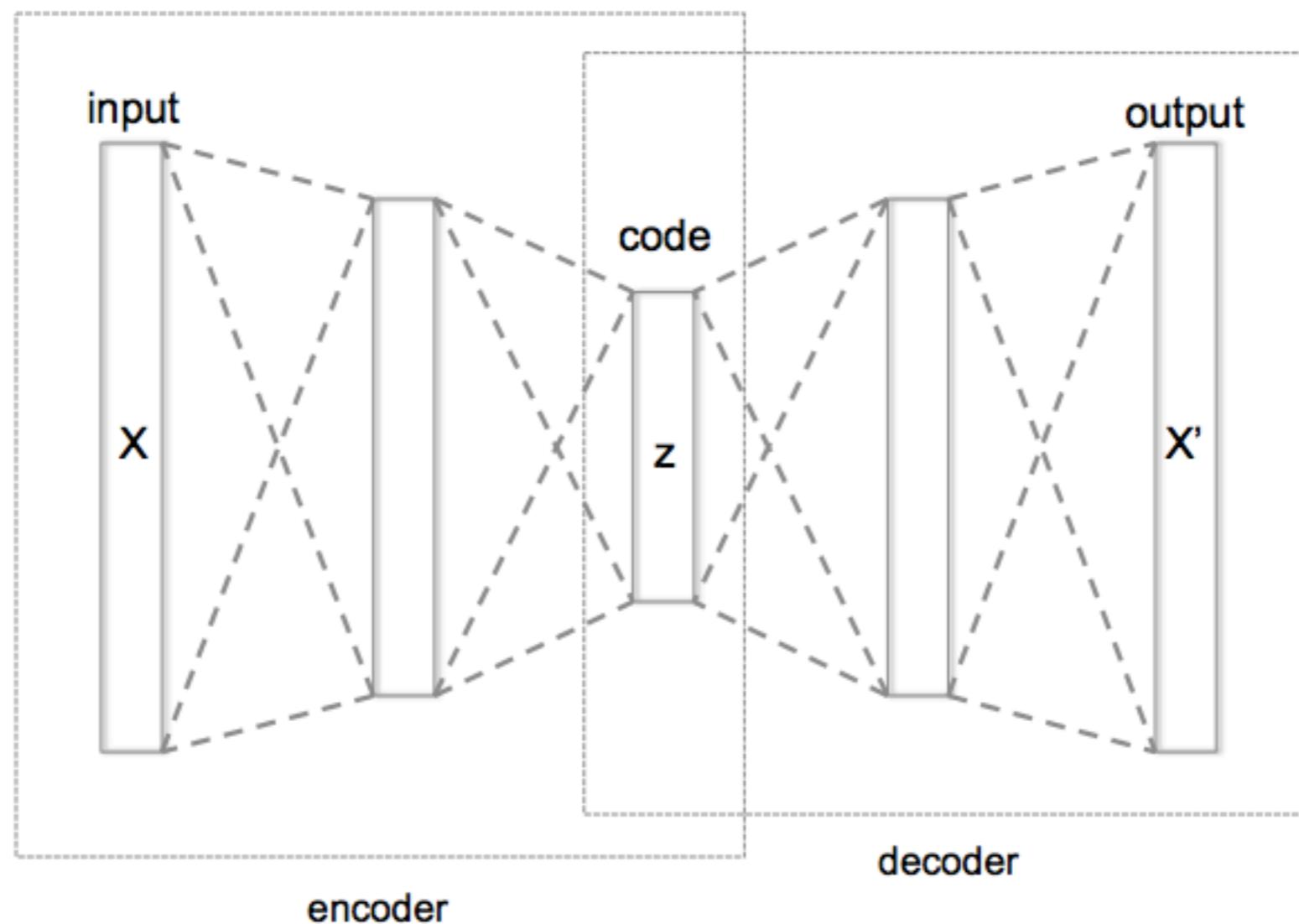
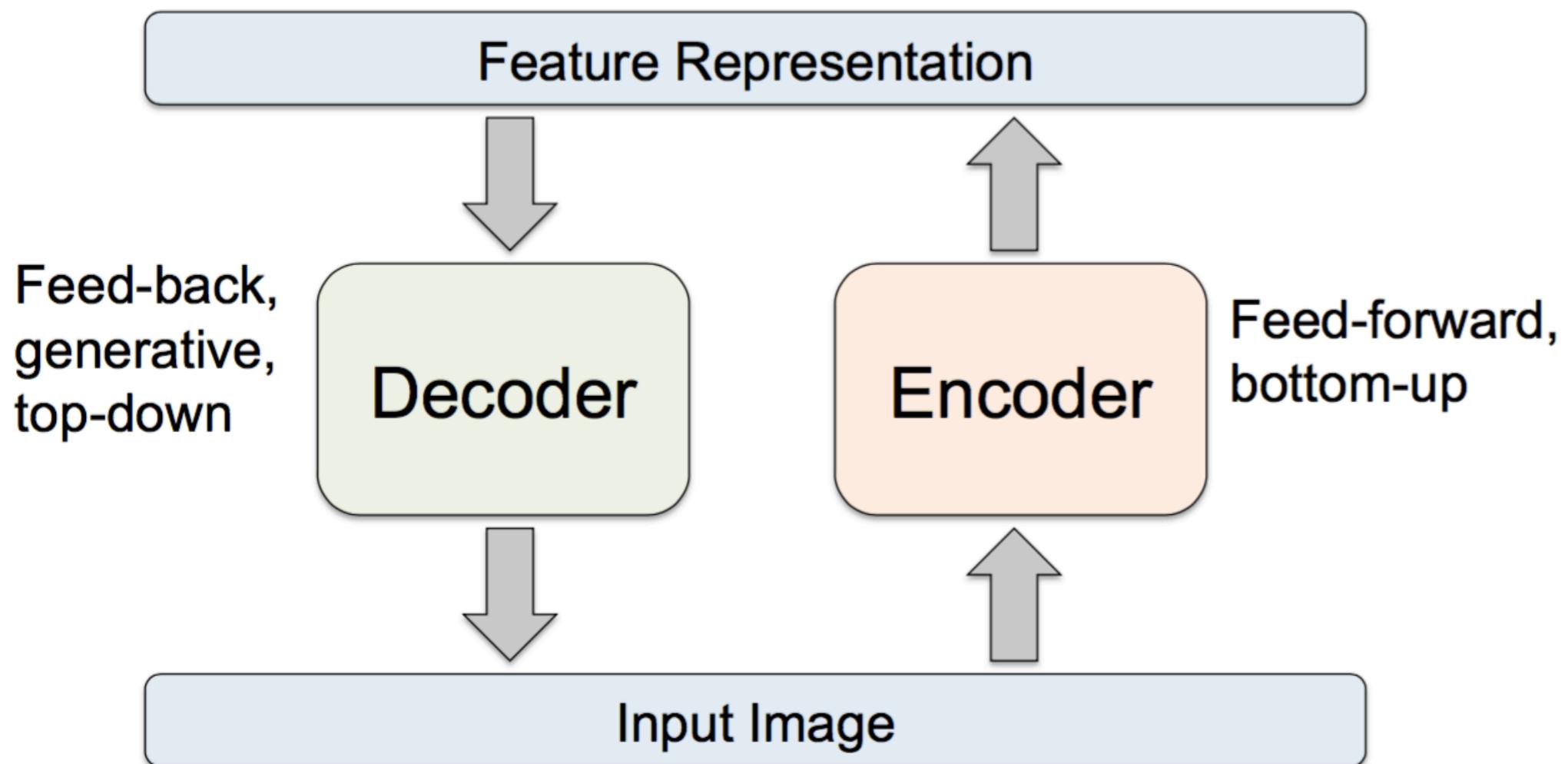


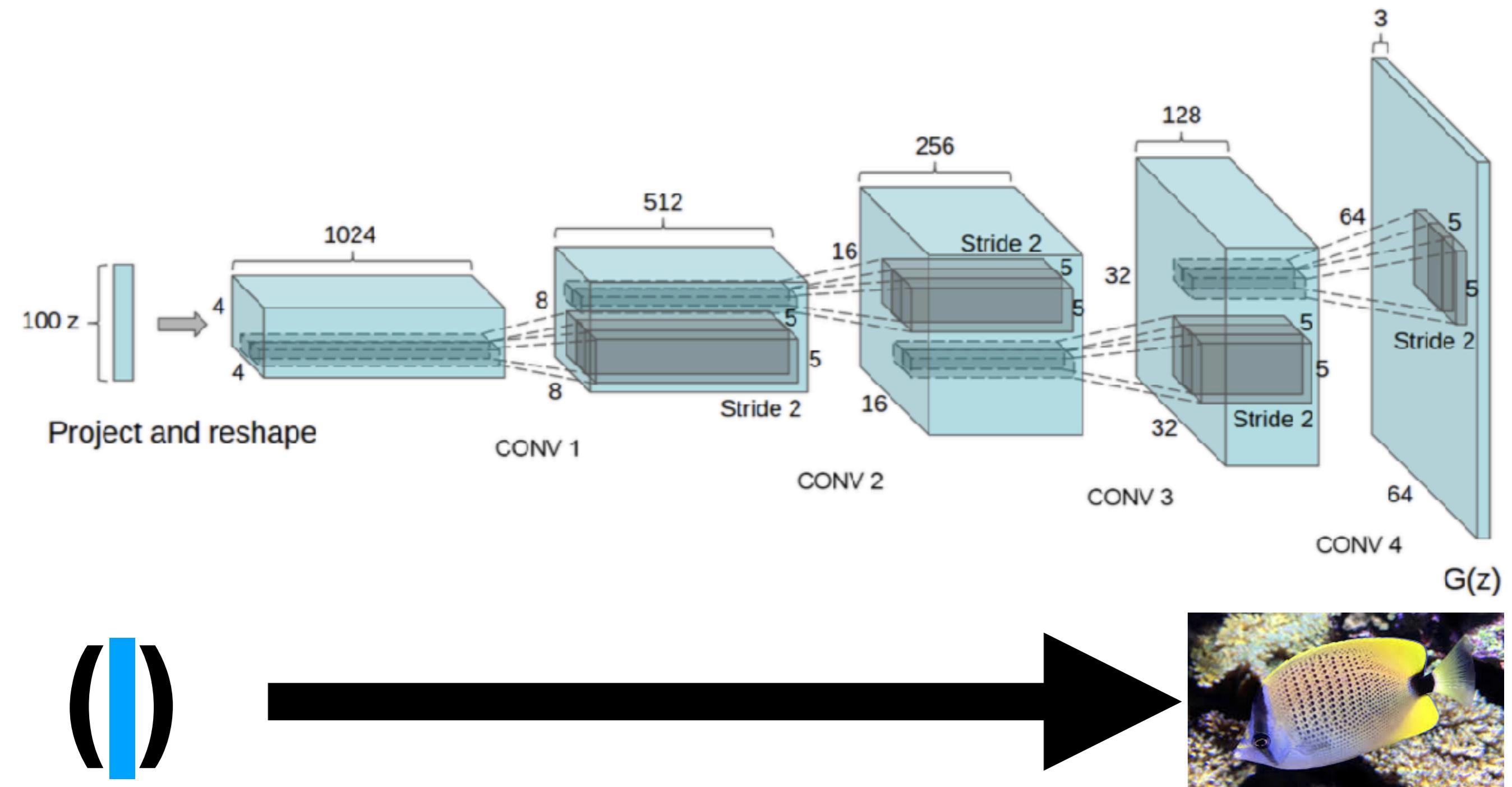
Figure from wikipedia

Autoencoders



- Details of what goes inside the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Generative Models (e.g., GAN)



Generative Modeling



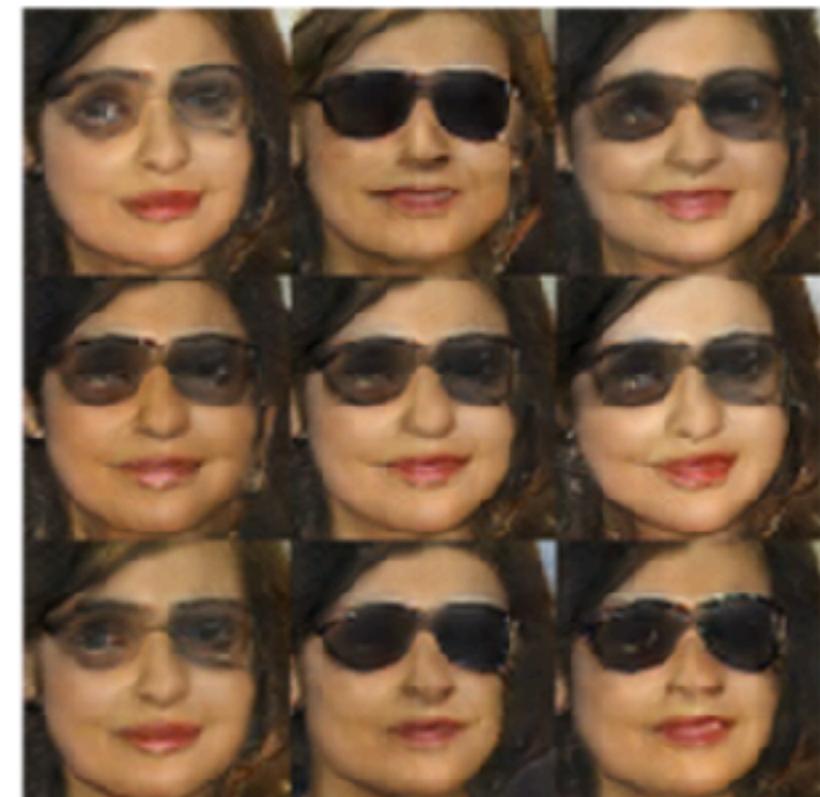
man
with glasses



man
without glasses



woman
without glasses



woman with glasses

NNs for ODE

$$\frac{dx(t)}{dt} = \sigma (y(t) - x(t))$$

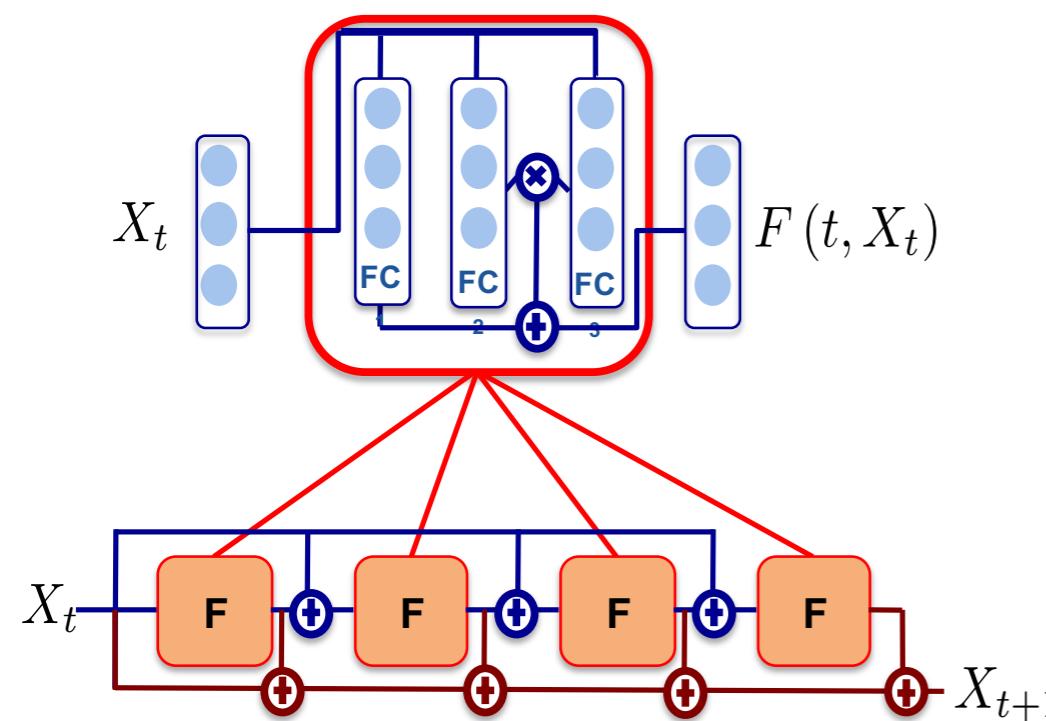
$$\frac{dy(t)}{dt} = x(t) (\rho - z(t)) - y(t)$$

$$\frac{dz(t)}{dt} = x(t) y(t) - \beta z(t)$$

Lorenz-63 equations

Two key steps:

- NN parameterisation for operator F
 $d_t X_t = F_\theta(X_t)$
- NN parameterization for the associated integration scheme



Adjoint operator:

- Adjoint of the forward integration scheme
- Backward NN integration of Adjoint of the forward integration scheme

Losses & training strategies

Synoptic (point-wise) losses

$$\arg \min_{\theta} \sum_i \|y_i - f_{\theta}(x_i)\|_p^p$$

Likelihood losses

$$\arg \min_{\theta} \sum_i (y_i - f_{\theta}(x_i))^t \Sigma_{\theta}^{-1} (y_i - f_{\theta}(x_i)) - 1/n \log |\Sigma_{\theta}|$$

Distribution-based cost (Optimal transport)

Generative Adversarial Networks

How to deal with situations where one cannot define an explicit criterion to train a generative model but numerous examples are available ?

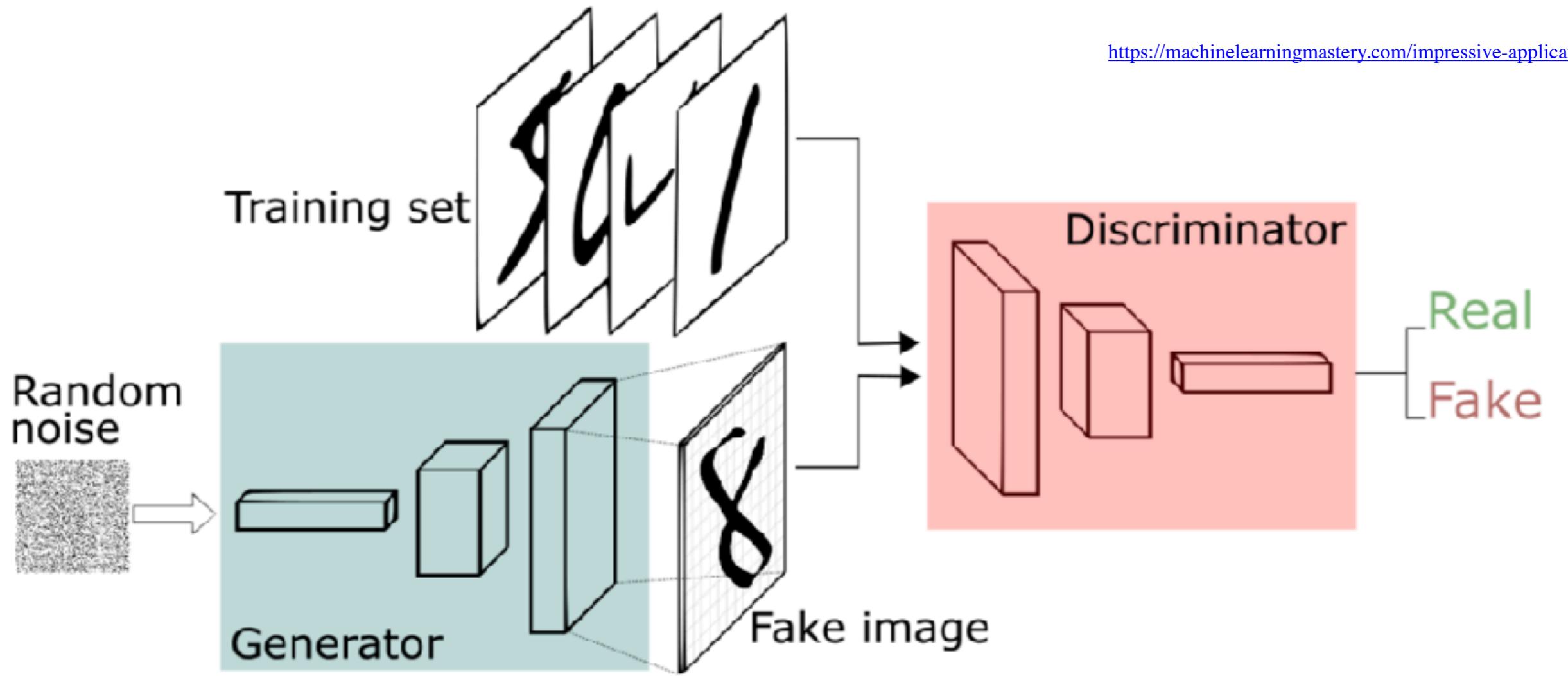


Image credit: Thalles Silva

Generative Adversarial Networks

Image processing examples

<https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>